ALTEGRAD challenge Fall 2020: H-index Prediction

Garde Joël
Telecom Paris, M2DS
joel.garde@telecom-paristech.fr

Abstract

We propose an end-to-end pipeline to solve a h-index regression problem using a co-authorship graph and authors' s papers abstracts. In particular, we handle both textual and graph data sources in a semi-supervised setting, for a relatively large dataset.

1 Overview of the architecture

I use features extracted in an unsupervised fashion from the textual data as input features to graph based deep-neural network trained directly on the regression task using the MAE loss.

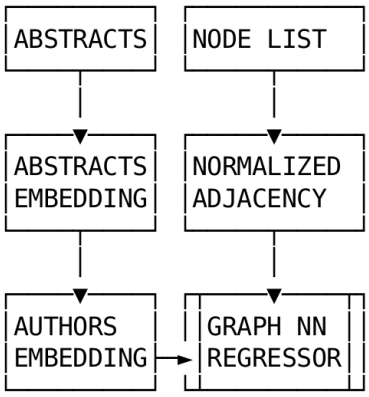

Figure 1: Flow diagram of the proposed architecture.

2 Preparatory work

A first pass is done to assemble the different sources. Authors ids are extracted from the edgelist and the author-to-abstracts mapping. The authors ids are mapped to integer keys to ensure the correspondence between rows in the adjacency matrix of the graph and authors ids.

3 Textual data pre-processing

motivation. Textual data is given as an inverted index. I wish to provide embedding for abstracts, by considering abstracts as bag-of-words. Additionally, the file size is sufficiently big to hinder exploration and trials. I reduce significantly the cost of further computations by providing a sparse representation of the texts.

parsing. I use regular expression to match ascii words of lengths 3 or more. There are non-Latin characters in the dataset, but not enough for them to have a meaningful embedding. This has the benefit of skipping the json deserialization step. It also handles the removal of punctuation and extra space characters present in the inverted index keys. Values of the inverted index, IE position of the word in the abstract, are dropped in accordance to the bag-of-word representation: an abstract is only represented by the set of tokens it contains.

stopwords. To lower the size of the dictionary, I remove stopwords. Stopwords were manually compiled from the list of the most frequent words in the dataset. Common phrasing in abstracts, such as *"We propose", "In this paper"* is thus removed. This make use of domain-specific information to lower the computational burden down the chain.

lemming. I took inspiration from the [Porter,] lemming algorithm and implemented step 0, the removal of final *s*. It is really fast to perform, this is the main raison over using the full lemming algorithm.

vectorization. Vectorization is done on the fly by associating to each word its number of appearance. A vocabulary is kept for verification purposes.

sparse representation. vectorized abstracts are concatenated together. Access to a specific abstract is done through an offset array. This is similar to a **CSR** representation in scipy.

4 Document and author embedding

Using the vectorized abstracts, I wish to provide embedding for each abstract.

4.1 model.

I use the Distributed Bag of Words version of Paragraph Vector (PV-DBOW) architecture from [Le and Mikolov, 2014]. Training is performed by using abstracts ids as inputs and solving a discriminative task. Positive words (from the same abstract) and negative words (from other abstracts) and the model learns to distinguish between them.

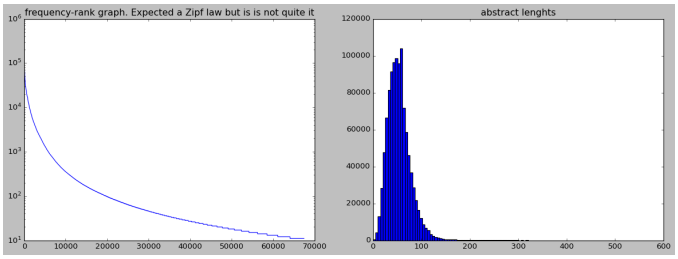

Figure 2: Textual data statistics. The log-frequency graph is too curved to be a zipf law. Texts lengths from a slightly skewed gaussian.

The model consists of two embedding matrices, $M_{doc} \in \mathbf{R}^{n_d \times n_h}$ and $M_{word} \in \mathbf{R}^{n_v \times n_h}$ where n_d, n_v, and n_h are the cardinalities of the abstract set, the words set and the embedding dimension respectively.

$S = M_{doc} \cdot M_{voc^T}$ is a similarity matrix. It encodes which words are compatible with which abstracts. Learning M_{doc} and M_{voc} using stochastic gradient descent is akind to an implict factorization of S as noted by [Levy and Goldberg, 2014], although I did not pursue this point of view further.

By using the PV-DBOW architecture, embeddings are learnt directly for the abstracts. Since the goal of this step is to provide features for authors, there is no need to learn word embeddings.

sampling. Negative samples are drawn in accordance with the blackout method [Ji *et al.*, 2016]. They proposed a loss function that takes in account the non-uniform sampling of the negative words. I use the experimental frequencies computed in the pre-processing step to define a unigram distribution Q where sample are drawn from.

implementation. The model was built using the pytorch [Paszke *et al.*, 2019] framework, in order to benefit from the parallel capabilities of GPUs. The sparce CSR representation of the abstracts is used to define an efficient data pipeline for the GPU. Model was trained for an hour, until loss seemed to stabilise. I choose n_h, the embedding dimension to be 100. Abstracts are rather short, and an embedding dimension that is not too high should help not overfit.

4.2 author embedding.

Given abstracts embeddings, I use the provided author-to-abstract mapping to form a set of embedding for each author.

4.3 Assembling abstracts

I choose to use a simple sum architecture defined as $e_{author} = \sum_{\mathcal{G}} e_{abstracts}$ where $\mathcal{G}$ is the set of abstracts associated with this particular author. This approach is motivated as a the sum is invariant to permutation, yet it retains information about the number of papers an author has.

It is important to retain information about the number of papers an authors has written. It has a high correlation with the h-index. In [Xiao *et al.*, 2016] work of predicting citations, both the h-index and the number of written paper are high ranked covariates.

I avoided introducing a full deepsets [Zaheer *et al.*, 2018] architecture to reduce complexity, but this could be tried later.

missing values. Some abstracts in the author-to-embedding mapping are not present in the embedding set. Those are replaced by zero valued vectors. Authors without any valid abstracts are given a zero-valued embedding too.

5 Graph semi-supervised learning

5.1 motivation

The provided edge-list encodes a collaboration graph between authors. It is a significant part of the provided data.

We also are provided a relatively small training set, I wish to use an architecture making use of graph features to learn a regression against the h-index in a semi-supervised.

5.2 graph convolution network

I used the graph convectional network architecture (GCN) of [Kipf and Welling, 2017]. They specifically designed this architecture in the semi-supervised context. it is defined as

$$Y = softmax(A\ relu(AXW_1)W_2)$$

where A is a normalized adjacency matrix with added self-loops.

By multiplying by A, the networks operates a message passing operation, where features from nodes are propagated to neighboring nodes. This message passing smooths out labels in the graph, making use of its structure.

5.3 node feature

An advantage of this architecture is that it readily admits vector valued features for each node. I tried multiple settings.

uniform. By using a vector of ones as the input feature of the GCN, I was not able to do better than 6 on validation Mean Absolute Error (MAE). Under this setting, I tried multiples depths and hidden dimensions with no success.

One-hot-encoded. On the other end of the spectrum of inputs, I tried using one-hot-encoded inputs. However, even with heavy regularization on the weights and severely restricting the network size, it leads to sever over-fitting rapidly. I was not able to better the uniform features.

extracted

Motivated by a comment from [Kipf and Welling, 2017], I tried initializing features from features extracted from the graph. In particular I used the degree of the node and the number of papers in the author-paper-mapping.

I also added the k-core decomposition number inspired from [Sarigöl *et al.*, 2014], where they use different centrality measures to predict future citation from a co-authorship network.

Using those three features and the graph convolutionnal network, I was able to go just under 5 in MAE.

5.4 Other approaches

Using the features listed in the extracted paragraph, the performance of a GCN was in fact similar or worse than that of a simple Fully Connected Network (FCN). Further work should investigate whether initializing the node with more

features could help. In particular, I would like to try initializing nodes features from the embedding learnt using Deepwalk for instance.

This under-performance of GCN compared to a simple FCN led me to look for other architectures manipulating graphs.

5.5 Simple Graph Convolutional Network.

GCN is not the only method that can be though of as a message passing architecture smoothing features over the graph. The work of [Zhou *et al.*, 2004] can retrospectively be seen as a feature-less message passing architecture. They directly use the training labels as the messages. However, they work can help understand hyper-parameter in GCN.

Their method essentially iterates

$$F_{i+1} = \alpha S F_i + (1 - \alpha) Y$$

until convergence. Here, F is the matrix of features at the current iteration. Y is the one-hot-encoded matrix of class label, S a normalized adjacency matrix without self-loops and α a meta-parameter. Under this light, α is similar to the self-loop coefficient in the GCN architecture. It represents how much energy a node receives from others compared to directly from the targets. By setting a high α, we force the representation to be more smoothed out over the graph. Therefore it should be beneficial to treat the self-loop parameter of the GCN as a regularization parameter.

Finally, [Zhou *et al.*, 2004] approach does not allow for nodes features. Another view that keeps their simplicity, but allows for nodes features, such as the authors embeddings we computed, is [Wu *et al.*, 2019]. [Wu *et al.*, 2019] hypothesized that the non-linearity used in GCN is not essential. Therefore, instead of building multiple layers of a deep neural network, they directly multiply their features by the normalized adjacency with self-loops K times.

$$Y = S^K X \Theta$$

This approach is particularly interesting as it uses the adjacency during a pre-processing step, by pre-computing $S^K X$. Thanks to this, we do not need to use all the nodes during training, but only those for which we have a label. Training using this method is two order of magnitude faster than with GCN. This translate into additional test performance by letting more time to tune hyper-parameters and to let the training go as far as possible before it overfits.

6 Baseline

dummy model. A dummy regression that only learns a global bias. Obtained a 7.20 L1 score on validation data.

FC model. A fully connected network using the degree and number of papers of as inputs. Obtained a 4.9 score on validation data.

7 Results

I use the simple GCN to solve the regression task on the h-index using the L1 loss directly. I keep 10% of the labeled dataset as validation.

A classifier model targeting the first 200 integers did not perform well. since the model outputs are real number and h-index are integers, the regressor's outputs were rounded to the nearest integer, with marginal improvements. Results were obtained using 128 hidden units, the Adam optimizer with 0.01 learning rate and 0.008 weight decay, trained for 500 epochs. Weight decay proved to be important, as even with the graph-smoothing, authors embeddings were sufficient to cause sever over fitting if not controlled for.

This model obtained a 4.7 validation score.

8 Discussion

As of the writing of this report my model performance is sub-par on the public testing leader-board. the model tends to over-fit when trained with the authors embeddings, even when using a graph-network to smooth it out.

Maybe a better usage of the graph would have been to extract more unsupervised features from it in addition to the degree, but I got discouraged when using the k-core number as a feature did not improve results.

This tendency to overfit using graph features could point in the direction of constraining more the embeddings in the unsupervised phase. Maybe by using a really low (10) embedding dimention.

References

[Ji *et al.*, 2016] Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. Blackout: Speeding up recurrent neural network language models with very large vocabularies, 2016.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[Le and Mikolov, 2014] Quoc V. Le and Tomás Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014.

[Levy and Goldberg, 2014] Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 2177–2185, Cambridge, MA, USA, 2014. MIT Press.

[Paszke *et al.*, 2019] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[Porter,] Martin Porter. The english (porter2) stemming algorithm. http://snowball.tartarus.org/algorithms/english/stemmer.html.

[Sarigöl *et al.*, 2014] Emre Sarigöl, Rene Pfitzner, Ingo Scholtes, Antonios Garas, and Frank Schweitzer. Predicting scientific success based on coauthorship networks. *EPJ Data Science*, 02 2014.

[Wu *et al.*, 2019] Felix Wu, Tianyi Zhang, Amauri Holanda de Souza Jr. au2, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks, 2019.

[Xiao *et al.*, 2016] Shuai Xiao, Junchi Yan, Changsheng Li, Bo Jin, Xiangfeng Wang, Xiaokang Yang, Stephen M. Chu, and Hongyuan Zhu. On modeling and predicting individual paper citation count over time. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 2676–2682. AAAI Press, 2016.

[Zaheer *et al.*, 2018] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov, and Alexander Smola. Deep sets, 2018.

[Zhou *et al.*, 2004] Dengyong Zhou, Olivier Bousquet, Thomas Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems*, volume 16. MIT Press, 2004.