Predicting the Academic Success using Multi-Faceted data and Limited-Supervision

Ashraf GHIYE[1] , **Hussein JAWAD**[1]
[1]DS - Ecole Polytechnique
Team: **Hear Me Roar**
{ashraf.ghiye, hussein.jawad}@telecom-paris.fr

Abstract

The aim goal of this final project[1] is the prediction of the academic success using multi-faceted data and limited supervision. Based on a set of top cited papers for each author and a network of co-authorship, we want to predict author's h-index. We explored three methods, first feature engineering and application of a regression model, semi-supervision and pre-labelling using Graph Convolutional Networks and finally Multi-Modal neural networks joining deepsets and meta-data. This report concludes the class of **Advanced Learning for TExt and GRAph Data** 2020.

1 Introduction

Since Hirsch's first publication of the h-index in 2005 (Hirsch, 2005) [10], this new measure of academic impact has generated a widespread interest.

The h-index is defined as follows:

A scientist has index h if h of his/her N_p papers have at least h citations each, and the other $(N_p - h)$ papers have no more than h citations each.

The advantage of the h-index is that it combines an assessment of both quantity (number of papers) and quality (impact, or citations to these papers) (Glänzel, 2006) [8]. An academic cannot have a high h-index without publishing a substantial number of papers. However, this is not enough. These papers need to be cited by other academics in order to count for the h-index.

Having said that, it is clear that for any successful attempt to predict an author's h-index we would need two types of data: **text data that assess the quality** of his publications (based on NLP-analysis) and **graph data that quantify his publications and collaborations** accross committees.

There has been works in literature trying to forecast the h-index of a given author [1], to measure the impact of a new paper on his current h-index [17], or to predict his future citations [14]. However, very little literature worked on combining graph and text data in such a task. It is our goal then in this project to deliver such a model that can predict the h-index for a given author based on the abstract of his top-10 cited papers and on a co-authorship graph of all the authors.

[1]https://www.kaggle.com/c/altegrad-2020

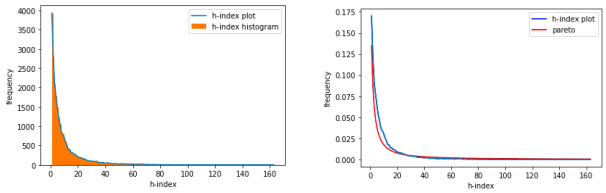

Figure 1: h-index distribution

2 Challenges

The aim of this section is two-folds, first to inspect the distribution of h-index to get a closer look on this quantity of interest. Then to present the challenges we might face and propose a way to move forward.

First, the quantity of interest is a discrete number characterizing each author. globally it has to fall naturally in the interval $[0, n_{max}]$ where n_{max} is the total number of papers ever published for a single author. It it is then a regression task with an integer target variable h.

It is important to ask if the all of these data at our hand, i.e. the predictors x does have the predictive power and can be used to predict h. Indeed we only have partial information that helps predicting h-index, other quantities such as the journals he published in, total number of citations for an author, author's age (career length) etc.. Nevertheless, we can extract useful information from the data at hand, for example, using document classification and clustering we can detect the fields and topics an author has worked in, from the graph we can determine his research committees and his links and influence with other authors. Many of these quantities showed a strong mutual information with the target variable h which proves their predictive power.

Second, the target variable is highly skewed. The majority of authors has an h-index below 20 with very rare extreme events. it is positively skewed as seen in Figure.1. This will cause some difficulties for algorithms to learn such a distribution since it has a lot of extreme values which are rare and underrepresented in the population. We must thus, include some statistical studies to accompany machine learning and we cannot go blindly with plug-in models.

Set	#papers	%papers
Training	174,660	10.02%
Test	1,569,211	89.98%
Total	1,743,871	100%

Table 1: Number of papers

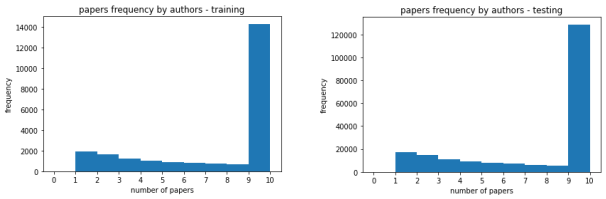

Figure 2: Papers/author distribution

Third, only 23,124 out of 231,239 authors are labelled with their h-index. Thus we have a limited-surpervision with only 10% of data annotated. This will require us to search around semi-supervised methods and leverage all the unsupervised algorithms we can apply.

Finally, we are faced with two type of data, texts data for which we have to get it through NLP pipelines to extract meaningful predictors such as topics, innovative keywords, and embeddings for a given author. We want to leverage the graph data as in which we can measure the influence and impact of authors in their network. Also, combining these two type of information into a single model can be done optimally such that they complement each other.

3 Data

To build the model, we are given two types of data:

- The abstracts of the top-cited papers of each author.
- Co-authorship graph that models the collaboration intensity of authors.

3.1 Text Data

Each author is represented with a list of his top-cited papers, this list does not exceed 10 papers and is not less than 1 paper for each author. Table.1 represent the total number of papers in each set.

Also, we made a little bit of statistical exploration shows that on average an author has around 7 to 8 papers and the median is 10 papers/author. The distribution in Fig.2 shows that both training and testing sets have the same distribution of papers/author.

The problem here is that the text of **only 1,056,539 abstracts were given**, which counts roughly for 60% of the total number of papers. Thus, we are not capable of using text alone for our regression problem. Therefore we must turn into other type of features, i.e. graph-based features.

3.2 Graph Data

The next type of data we have is a co-authorship network where nodes correspond to authors that have published papers in computer science venues (conference or journal). It is an **unweighted graph**, i.e. two nodes are connected by an edge if they have co-authored at least one paper. The graph consists of 231,239 vertices (total number of authors) and 1,777,338 edges in total.

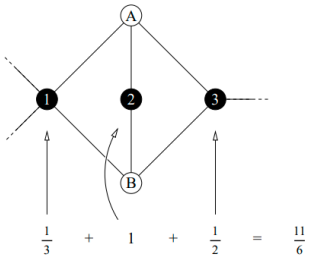

Figure 3: Authors A and B have coauthored three papers together, labeled 1, 2, and 3, which had respectively four, two, and three authors. The tie between A and B accordingly accrues weight $\frac{1}{3}$, 1 and $\frac{1}{2}$ from the three papers, for a total weight of $\frac{11}{6}$.

This type of data will make a complementary for papers' abstracts at our disposal since text data can be seen as static representation of an author (i.e. his portfolio in research) and the graph data is a dynamic representation of him (i.e. his position and interactions in the research community).

The original network has been extracted from Microsoft Academic Graph[2].

However, knowing the list of papers for each author we have inverted the dictionary to extract for each paper a list of authors who worked on that paper. From this information we built another **weighted graph**, where each two authors are now connected with an edge of weight equal to the number of time they collaborated.

The first weighted graph is denser, i.e. it has more edges than its unweighted counterpart, which suggests that the unweighted graph had only partial information, for example it can be that the graph is out-dated and authors had new papers and their collaboration is not updated in the graph. But also we found that some edges were only present in the original graph, this is again due to the fact that we only had partial information about the papers and not all of them are present (these papers might be omitted).

We thus started from the original graph were we kept the original edges untouched and added new edge with weights proposed in [12] whenever information about the frequency of collaboration was available. These weights are computed as follow and showed in Fig.3:

$$w_{ij} = \sum_k \frac{\delta_i^k \delta_j^k}{n_k - 1}$$

where n_k is the number of coauthors of paper k and we explicitly exclude from our sums all single-author papers (They do not contribute to the coauthorship network, and their inclusion would make w_{ij} ill-defined). This weighting schema will make authors who worked on a paper with few collaborations have a stronger connection between them and we will

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

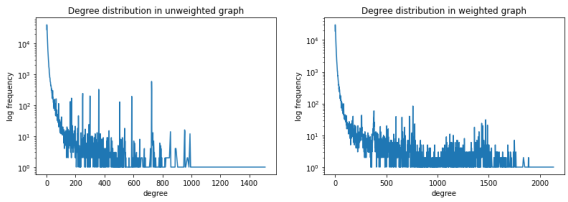

Figure 4: degree distribution in collaborative network

sum the connection between two authors as the sum of their normalized collaboration defined above[12].

Finally, this crafted graph of 2,982,240 edges encodes the frequency of collaboration between authors and the strength of their bonds. The degree distribution in Figure.4 confirms the property of large real-world network, the degree distribution indeed follows a power law. Also, the unweighted graph is a bit noisy which might be due to the fact of partial information before adding the complementary information about collaboration.

4 Features

4.1 Text-based

4.1.1 Text Embeddings

In this section, we try to create a vector representation of the properties for each author based on what they have written so far. To do so, we focus on creating a document embedding model that represents whole paragraph and their semantic information as vectors. This helps the machine understand the context, intent, e.g. author's field of research, and other nuances throughout the text such as keywords and buzzwords that can indicates the quality of this paper. Then, we apply aggregation functions (sum, mean, median) on all the articles of an author, in order to obtain a global representation of his writings and his field of work.

A sentence or document embedding is an interesting area of research in natural language processing (NLP), a subject that is still actively explored for its various features. In general, there are four well-known techniques that we can use to get sentence embedding: Doc2Vec, SentenceBERT, InferSent, Universal Sentence Encoder.

In this paper, we focus on only two of these four techniques: Doc2vec and SentenceBert.

Doc2vec

Introduced in 2014 [11], Doc2vec is an NLP tool for representing documents as a numerical vector and is an extension of Word2Vec. It is an unsupervised algorithm that generalizes the Word2Vec model by introducing another "paragraph vector" to capture a global semantic of the whole documents as well as an aggregations of its words embeddings. Word2Vec architecture can be implemented using two algorithm: continuous bag of words (CBOW) or skip-gram (SG), while in the Doc2Vec, the corresponding algorithms are distributed memory (DM) and distributed bag of words (DBOW).

We used Genism library which implements a DBOW's algorithm for Doc2Vec to extract a **64**-dimensional representation for each paper's abstract. We then represented each author as an aggregation, namely a sum of his papers. The aggregation function was selected based simple baseline.

SentenceBERT

Introduced in 2018 [13], SentenceBERT is a BERT-based model that uses Siamese and triplet network structures to derive semantically significant sentence embedding that can be compared using cosine similarity. The Siamese network takes 2 sentences as input, passes them to a BERT model and a pooling layer, and then calculates the cosine similarity of the vectors from the pooling layer.

We tried this model because in our case we have each paper represented as a short paragraph of 200 words on average, and we want to catch the similarity between papers based on their contents (often two papers with the same quality or in a certain fields both tend to have high citation and impacts). We wanted to go further and fine-tune these models on the h-index but we faced many problems due to computational power and time constraints.

4.1.2 Topic Modelling

Topic modeling aims to identify topics that appear in a collection of documents using a statistical or unsupervised learning approach. We hypothesize that trending topics such as machine-learning might attract greater interest and thus potentially increase the citation of related articles, meaning that authors dealing with trending topics will be cited more often and thus increase their H-index. This observation encourages us to group and identify related articles and then assign them a number representing their topics. We can therefore predict each author's topic by applying aggregation functions such as (max occurrence, diversity of publications, etc..), and in terms of training, the model can predict whether the author is dealing with a interesting topic or not.

Clustering

We want to ensure that papers with related topics are grouped together. However, many clustering algorithms handle high dimensions poorly, so we need to reduce the dimension of the embedding space before we apply clustering. We thus used **UMAP** which is one of the most widely used algorithms for dimensionality reduction. It works very well for clustering because it preserves a significant part of the local high-dimensional structure in the lower dimensions.

4.2 Graph-based

In this section we focus on features extracted from the co-authorship graph. Since the dynamic of interaction between authors is encoded in a collaborative graph, it is thus necessary to leverage this information in our prediction of h-index since it is highly affected by committee and author's network.

Below, we list the main features extracted from both unweighted and weighted graphs.

4.2.1 Centrality Measures

Prior research provides evidence that leading scientists, innovators, etc, occupy structurally significant locations in collaboration network graphs. Since centrality measures are frequently used to describe a node's relative importance within a network graph, then such high-value individuals should be discover-able using network analysis techniques.[3]

- **Author degree**: node degrees from unweighted graph. This feature measure the number of authors a researcher has worked with. It reflects his network size and communication potential within the network.
- **Core number**: to help identify if an author plays a central role in his cluster (committee) and if he makes part of a denser sub-graph that has a consistent work together.
- **Betweenness**: it is regarded as a measure of a node's control over communication flow. It is helpful in our case since author who is regarded as a connector between two different committees will probably have more publications and citation.
- **PageRank**: it provide a link-based analysis that estimates a the author's importance based on the importance of other authors that are connected to him.
- **Clustering coefficient**: it measures the fraction of possible triangles through that node (i.e. author) exist, meaning it measures how many time two authors v and w who have worked with author u have also worked together.
- **Eigenvector centrality**: it measures the influence of a node in a network. A high eigenvector score means that a node is connected to many nodes who themselves have high scores, i.e. an author is connected to influential network and they probably have high h-index.

4.2.2 Node Embeddings

Our goal is to obtain a low-dimensional representation of each author such that similar authors would be close in the embedding space. Similarity of nodes will be based on their structural similarity (the role they play in their committees) and their neighbours.

There are various node embedding techniques existing in literature, most of them are unsupervised techniques which is helpful in our case. Stanford's tutorial[15] gives a complete tour on them. However, we were unable to test all of them, mainly due to limited time and space constraints. We tested out two Shallow embeddings techniques: DeepWalk[4] and LINE[5].

DeepWalk

The main idea is to to encode random walk's statistics in learning of embeddings, meaning that dot product of author u and author v in the embedding space must reflect the probability that u and v co-occur on a random walk over the network.

These latent representations encode social relations in the collaborative network in a continuous vector space and learns a representation which encodes structural regularities. We trained the algorithm on the original unweighted graph to learn a **64**-dimensional vector representation for each author.

Finally, we believe that Node2Vec[9], which is an improvement on DeepWalk with the same objective of encoding deepwalks statistics but using biased-walks instead, will give better results. Node2Vec is able to capture both macro and micro information about committees by controlling p and q, the parameters for BFS and DFS. Unfortunately it needs more tuning and computations which were not available at that time.

LINE

Empirical experiments proved the effectiveness of LINE on a variety of real-world information networks, including citation and collaoration networks.

The method optimizes a carefully designed objective function that preserves both the local and global network structures. We trained the algorithm to extract 8-dimensional representation for each author.

4.3 Hand-crafted

Feature engineering is an important part in every traditional machine learning approach. Since our regression models will contain decision-tree based algorithms, it is important to craft some features that reflect sums and aggregations since these former algorithms unlike linear regression models are prone to learn this kind of information, i.e. learning sums and ratios between features is not possible in a decision-tree based regressor.

- **Papers count**: for each author, how many papers does he have in total, this number is upper-bounded by 10 (due to the limited list of 10-top cited papers) and lower-bounded by 1.
- **Abstract statistics**: for each authors, the min, max, mode, median and average word counts of his provided abstracts.
- **Topic statistics**: after clustering each paper in one of 32 possible clusters that can be interpreted as 32 different topics, we represent each author as a vectors of size 1 to 10 topics. We thus extracted the mode (most frequent topic) and percentage of contribution (number of papers in this topic divided by his total number of papers).
- **Diversity**: also based on topic modellings, an author who works in various discipline and publish in many topics is more able to have a higher h-index, thus we counted the unique number of topics an author published in.
- **Neighbours' information**: many times an author success can be measured and identified based on his collaborators. Thus, we used second-order measures, i.e. for each author we took the min, max and average of his neighbour's degree, PageRank and number of papers.

5 Prediction Models

5.1 Traditional Regressors

h-index prediction can be considered simply as supervised regression task. Thus as a baseline we will solve this mainstream task using traditional machine learning methods. We can stack the nodes' feature vectors extracted in section 4 into a 2D array X, known as the design matrix, train some of sklearn's[3] regression models and evaluate their predictions.

These models are listed below and are ordered by increasing complexity, beginning with simple models to baseline our work, and ending with state-of-the-art machine learning algorithms.

[3] https://scikit-learn.org/stable/supervised_learning.html

(1) **Lasso**: Regularized linear regression model (sparse regression) with L1 penalty. The interest of this model is learning the features which are of the most importance in the prediction of h-index, thus for further analysis it enjoys high-interpretability.

(2) **KNN Regressor**: K-nearest neighbours, with number of neighbours k = 9. Also, heuristically author's success often can be measured by averaging the success of the most similar profiles.

(3) **Random Forest**: Ensemble regression trees using randomization techniques to improve performance.

(4) **Gradient Boosted Regression Trees**: a collection of simple regression trees that are trained iteratively by a type of functional gradient descent. We tried Lightgbm, Xgboost and Catboost, results show that Lightgbm outperformed all the other variants.

We will consider two tasks, regression of h-index directly and the prediction of its **log-transform** (h-index can then be obtained by the inverse transform $10^{y_{pred}}$). It is because the distribution of h-index is highly skewed in the original scale, logs "pulls in" more extreme values on the right (high values) relative to the median, while values at the far left (low values) tend to get stretched back, further away from the median. More details about log-transforms regression in [2].

5.2 GNN, GCN and GAT

The traditional approach won't exploit the full relationships between authors, it's performance will be determined by the quality of hand-crafted features. However, the above method ignores the collaborative relationships between the authors. We believe that these relations provide additional information about the similarity and the roles between authors and thus are essential in determining his influence and h-index.

Calculating quantitative values of the structural position of a node in a graph such as a node's degree or its pagerank can yield good results. However, what we have learned from the success of deep learning and convolutional neural networks in particular is that these algorithms are adept at automatically learning essential features that maximise performance of a downstream task.

Graph Convolutional Network (GCN) is one of the most successful neural networks for graph data. It works by smoothing a node's feature vector based on those of its immediate neighbours in the graph. In its article, GCN originally[7] was used to handle semi-supervised node classification.

It is our choice because of the limited-supervision nature of our problem, only a small number of authors are labelled with their h-index. Thus to adapt GCN to our task, we designed a new task, which is clustering or classifying each author based on their h-index.

- **Class 1**: authors have h-index ≤ 33 (95% quantile).
- **Class 2**: authors with h-index $\in]33, 61[$ (between 95 and 99 % of distribution).
- **Class 3**: authors with h-index more than 61 (1% of extreme cases).

We have performed the **semi-supervised node classification task using the unsupervised Deep Graph Infomax algorithm to pretrain a 2-layer GCN model with 32 and 16 hidden units** resp. using open source StellarGraph[4] library.

Although we could have used more classes to aggregate authors into more buckets (e.g. first 25, 50 and 75 percentiles) we only wanted to provide a demo because of the short time for this project. Nevertheless we believe that this pre-labelling task will play a central role in detecting extreme values (top tier authors) which are causing the most error in prediction. But more tuning and time is needed to explore this path thus we left it as a future research direction for us. We also explored GAT networks in the same task, all these details will be provided in a notebook alone and further results will be omitted from this report.

5.3 Multi-Modal Neural Network

Finally, our task consist of combining two different type of information: the abstracts (texts) and interactions (graph) of authors. It is natural then to consider a multi-model (MMNN) architecture to handle the regression task 5.

An author is presented as a **set of 10 of his top-cited papers' embedding**, thus we will use DeepSets[6] architecture to apply a series of transformation on this set and consider the aggregation of them as the first hidden vector representation of him, namely v_1.

Note that DeepSets is suitable because the h-index is assumed here to be independent on the arrangement and order between papers, i.e. our task does not have time-series nature and we are predicting a static quantity based on the collection of his papers. However, it is important to have only the papers published before labelling his h-index and omit the ones who came after but we don't have such information provided.

Also, the author can be expressed with the hand-crafted features extracted from graph in the previous section. this **meta-data** will pass into a 2-layer MLP to form a second hidden vector v_2.

After concatenating the two representations we will apply another 2-layer MLP with ReLu activation to predict the h-index of an author. We will then back-propagate with L1 loss to optimize the network and learn **jointly** both models.

6 Experimental Results

To evaluate the performance of our predictions we consider the MAE - Mean Absolute Error - metric described below. MAE can be easily interpreted and reflect the error in the

[4] https://stellargraph.readthedocs.io/en/stable

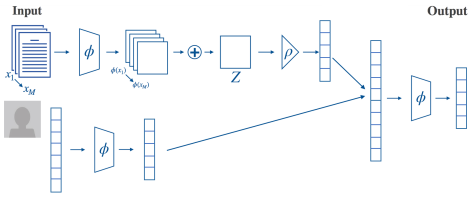

Figure 5: Multi-Modal simplified architecture

Table 2: Experimental results and MAE scores

	h-index		log h-index	
Model	Train	Validation	Train	Validation
Lasso	4.37	4.61	3.74	3.95
KNN	5.96	6.43	3.74	3.95
RF	1.45	4.04	1.53	3.85
LGBM	1.03	3.65	1.04	**3.20**
Xgboost	2.11	4.05	2.03	3.77
MMNN	3.49	**3.43**	-	-

same unit of measure of the quantity of interest, i.e. an MAE of 3 for example means an average error of +- 3 on output's prediction.

$$MAE = \frac{1}{N}\sum_{i}^{N} |h_i - \hat{h}_i|$$

where N is the total number of authors, h_i is the h-index of author i and $\hat{h}_i$ is the prediction for author i.

To conduct our experiments we used google cloud virtual machines, specifically all these experiments was handled on a **E2-standard instance with 16vCPU and 64GB of ram**.

As we can in the Table.2 a log transformation on the target variables made a drastic improvement for all models. This is because the data is highly skewed, also another transforms such as inverse hyperbolic sinus performed as good as log. We haven't got the time to explore the multi-modal neural network on a log transform and tune the model again, but we believe it will outperform again in log scale confirming the trends in Table.2

7 Conclusions

To this end, we have presented our work on **ALTEGRAD 2020's challenge**. The objective is predicting h-index of authors based on the abstracts of their top cited papers and the collaboration network extracted from Microsoft Academic Research. On the surface the declared task seems an ordinary regression task. However, underneath it lies vast challenges: limited information about abstracts (40% documents were missed), data skeweness, multi-source data, limited-supervision and many more.

Since this challenge was held on Kaggle, we devoted a great deal of our time on feature extraction and feature engineering to finally apply LightGBM regressor to achieve high performance on the leaderboard because of the limited time to apply research axes to get good results immeadiatly.

However, we kept a good balance and researched many directions. Most of the done work is not listed here because of the limited space too. We will finally present brievely some research ideas in the following section.

7.1 Research Directions

We worked on many improvements during the pipeline, but our main focus relied on the following three topics:

- **Limited-Supervision**: we tried to use many unsupervised and semi-supervised techniques to obtain some sort of pseudo-labels before conducting the supervision regression tasks.
- **Multi-Faced**: because of the heterogeneous nature of our predictors variables we had to explore ways to combine mixed data and design of multi-modal networks.
- **Data Skewness**: because of the extreme values presented in the model, these samples cannot be neglected since they are are considered as extreme events and not noisy samples. Thus we tried many techniques: log-transforms, ISH transforms[16], output corrections, robust regressions...

Other inconclusive works have been done, some of them came in last minute and needed more time to assess their validity.

- **Extreme Value Models**: the idea is to have two models: the first is trained for ordinary authors by excluding the extreme cases, and the second on the 1% percentile which harms the model the most. For testing we would use semi-supervision techniques in pre-labelling authors belonging to one of two or three classes that can be seen as indicators of flags of his potential position on the distribution, then based on that passed to the corresponding model to predict his h-index finally. The idea comes after noticing that learning a model on the authors with h-index below 61 yields an MAE error of ≈ 1.
- **Data Augmentation**: another approach is to over-present the extreme authors in the dataset. This also come in favour of another problem which is limited annotated data. We tried to augmenting these authors by duplicating their meta-data features but with different set of papers embeddings as permutation of his provided papers.
- **Post-Processing**: we tried to fit the h-index using a Generalized Linear Model. Pareto achieved the best in approximating the distribution, thus we tried to apply histogram transformation (inspired by images processing techniques) on the output of predictions to stretch out values since our model was biased to predict low-range values.
- **More GNN**: we tried GNN networks whenever was possible, but we didn't explore enough their full capacity to handle this task because we focused on getting both research and kaggle's performance results equal.
- **Similarity Graphs**: Finally we tried to use cosine similarity between two author's based on their abstract embeddings to weight the edges and wanted to test KNN approach and label propagation approaches.

References

[1] Allesina S. Kording K. Acuna, D. Predicting scientific success. *Nature*, 489:201–202, 2012.

[2] Kenneth Benoit. Linear regression models with logarithmic transformations. 2011.

[3] John Cardente. Using centrality measures to identify key members of an innovation collaboration network. 2012.

[4] Bryan Perozzi et al. Deepwalk: Online learning of social representations. 2014.

[5] Jian Tang et al. Line: Large-scale information network embedding. 2015.

[6] Manzil Zaheer et al. Deep sets. 2018.

[7] Thomas N. Kipf et al. Semi-supervised classification with graph convolutional networks. 2017.

[8] W. Glänzel. On the opportunities and limitations of the h-index. *Science Focus*, 1(1):10–11, 2006.

[9] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. 2016.

[10] J.E. Hirsch. An index to quantify an individual's scientific research output. *arXiv:physics/0508025*, 2005.

[11] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[12] M. E. J. Newman. Who is the best connected scientist? a study of scientific coauthorship networks.

[13] Iryna Gurevych Nils Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. arXiv:1908.10084, 2018.

[14] Jie Tang et al. Rui Yan. Citation count prediction:learning to estimate future citations for literature. *CIKM'11*, page 24–28, 2011.

[15] SNAP. Representation learning on networks, 2018.

[16] Arthur C. Tsai, Michelle Liou, Maria Simak, and Philip E. Cheng. On hyperbolic transformations to normality. *Computational Statistics Data Analysis*, 115:250–266, 2017.

[17] Reid A. Johnson et al. Yuxiao Dong. Will this paper increase your h-index?: Scientific impact prediction. *WSDM '15*, page 149–158, 2015.