Scientometrics (2019) 120:87–104
https://doi.org/10.1007/s11192-019-03110-2

Predicting authors' citation counts and *h*-indices with a neural network

Tobias Mistele[1] · **Tom Price**[1] · **Sabine Hossenfelder**[1]

Received: 6 July 2018 / Published online: 4 May 2019

Abstract
We here describe and present results of a simple neural network that predicts individual researchers' future citation counts based on a variety of data from the researchers' past. For publications available on the open access-server arXiv.org we find a higher predictability than previous studies.

Keywords Neural network · *h*-index · Arxiv · Citation metrics

Introduction

Measuring and predicting scientific excellence is as daunting as controversial. But regardless of how one feels about quantifying quality, measures for scientific success are being used, and they will continue to be used. The best that we, as scientists, can do is to at least come up with good measures.

The attempt to quantify scientific success by the number of publications dates back to the early nineteenth century (Csiszar 2017), but the idea really took off with the advent of the internet when data for publications and citations became easily accessible. Since then, a large variety of different measures has been suggested. Dozens of variants exist of the Hirsch-index (Hirsch 2007) (hereafter *h*-index) alone, such as the *g*-index, the *hg*-index, the *h-b*-index, the *m*-index, the *A*-index, *R*-index, tapered-*h*-index, and more [for a review see (Alonso et al. 2009)]. Besides other ways to evaluate citations—such as using algorithms similar to Google's PageRank on citation networks (Samuel 2015)—there are measures based on entirely different data, for example download statistics, the connectivity of co-citation networks, or social media engagement. A brief review can be found in Van Noorden (2012) and a more complete survey in Waltman (2015).

Despite the variety of measures, those based on citation counts have remained the most widely used, probably to no small part because they are fairly straight-forward to calculate from bibliometric data. Because of their widespread use, the question whether citation indices in their various forms can be predicted for individual researchers has attracted quite

✉ Sabine Hossenfelder
hossi@fias.uni-frankfurt.de

[1] Frankfurt Institute for Advanced Studies, Ruth-Moufang-Str. 1, 60438 Frankfurt am Main, Germany

some attention. The interested reader is referred to eg Yan et al. (2012), Dong et al. (2014), Zhang et al. (2016), Nezhadbiglari et al. (2016) and references therein.

We maintain that the best way to assess a researcher's promise is to study their work in depth. But bowing to the need to have a tool for administrative and organizational purposes that is fast and easy to use and allows at least superficial evaluation, we here report on the training of a neural network to improve the predictive models of presently existing measures.

Due to the large variety of existing measures and their predictors, we will not compare the method presented here to all of them. We will focus in particular on three previous studies that put our work into perspective. This is (1) the original paper about the predictivity of the Hirsch-index (Hirsch 2007), (2) a 2012 study (Acuna and Allesina 2012) which used a linear regression model to predict the *h*-index for research in the life sciences, and (3) a 2017 report (Weihs and Etzioni 2017) on various machine learning models used to predict the *h*-index for a large cohort of authors in the computer sciences.

Our paper is organized as follows. In the next section we clarify exactly what we aim to achieve. In the third section we document which input data we have used and how we have encoded it. In the fourth section we explain how we set up the neural network, and in the fifth section we will present our results. We finish with a discussion and conclusion in the sixth section.

Aim

Before we build a neural network, we first need to make precise what we mean by "predictive" and how we will measure this "predictiveness."

Our neural network will be fed publication data (third section) for a training group of researchers during a first phase of publishing activity, hereafter referred to as the 'initial period'. The aim is then to use the neural network (fourth section) to predict individual authors' performance in a second phase of publishing activity, hereafter referred to as the 'forecasting period'. The input data for the initial period does not include citations from papers that were only published during the forecasting period. The prediction period starts at 1/1/2008 which is chosen such that we can evaluate the neural network's performance for predictions up to 10 years into the future of the initial period.

Once the network has learned forecasting from the training group, we make a forecast for the remaining researchers—the "validation group"—and evaluate how good our forecast was. This is to say, in this present study we do not make actual future forecasts because we want to assess how well out network performs, but our method is designed so that it could be used to make real forecasts.

We will use two different approaches to evaluate the forecasting performance in the validation group.

The first approach (see "Comparison with the *h*-index" section) follows the procedure used in Hirsch (2007), which evaluated the predictiveness of the *h*-index for various quantities in terms of the correlation coefficient, *r*. In this approach, we do not include citations from papers in the initial period in the number of citations to be forecasted in the forecasting period. In making this clean separation, we get a better grasp on predicting *future* scientific achievement as opposed to *cumulative* achievement.

For this first approach, we follow the notation of Hirsch (2007) and denote the number of citations received between t_1 and t_2, i.e. during the forecasting period, as $N_c(t_1, t_2)$.

Since it is known that the h-index roughly scales with the square-root of citations, we will more precisely use $N_c(t_1, t_2)^{1/2}$ to make it easier to compare our results with those of Hirsch (2007).

The second approach (see the "Comparison with earlier machine learning predictions" section) follows a different procedure which is better suited for comparison with the results of Acuna and Allesina (2012), Weihs and Etzioni (2017). For this, we feed the neural network the same input data as in the first approach but then predict the cumulative h-index after n years until the end of the forecasting period. Furthermore, in this second approach we use the coefficient of determination, R^2, instead of the correlation coefficient, r, to quantify the goodness of our prediction because the same procedure was followed in Acuna and Allesina (2012), Weihs and Etzioni (2017).

Unless otherwise stated, our general procedure is to employ 20 rounds of Monte Carlo cross-validation—i.e. redo the random split into training and validation data 20 times and retrain the neural network—and report the mean as well as the standard deviation. One reason for employing cross-validation is that different splits of our dataset into training and validation data lead to slightly different results as will be further discussed below. Another reason is that our dataset is not particularly large. Cross-validation then allows to avoid splitting our dataset into a training, a validation, and test group while still avoiding overfitting to a particular split into training and validation data.

We did not tune the hyperparameters of the neural network. We chose the batch size and the number of epochs to train once and never changed them (see also the fourth section). All other hyperparameters were left at their default values.

Data

We have obtained the publications for each author from the arXiv through the publicly available Open Archives Initiative https://arxiv.org/help/oa/index and corresponding citation data from Paperscape http://paperscape.org. Paperscape citations contain only references that can also be identified with an arXiv paper. The fraction of these papers is highly dependent on the field. In the fields that for historical reasons most dominate the arXiv (hep), the identification rate is as high as 80%, while in some categories that are fairly new to the arXiv, it may be as small as 5%. The categories with the low identification rate are however also the categories that overall do not contribute much to the sample. For details please refer to https://github.com/paperscape/paperscape-data/.

For the purposes of this present work, we consider only the arXiv publications in the 'physics' set, which gives us a total of 934,650 papers. Journal impact factors (JIFs) are taken from Clarivate Analytics (2017). We group together similar author names and treat them as a single author by the same procedure as laid out in Price and Hossenfelder (2018). The featurization of our data is described in more detail in "Appendix 1".

From the complete dataset, we select a sample of authors and trim it in various ways. First, we require that they published their first arXiv paper between 1/1/1996 and 1/1/2003. We have chosen that period to span the time between 5 and 12 years prior to the cutoff which matches the procedure of Weihs and Etzioni (2017).

We then remove authors who have published fewer than 5 or more than 500 papers to avoid statistical outliers which would unduly decrease the predictivity of our method.

Finally, we exclude large collaborations, since their publication activity differs greatly from that of individuals. For this, we remove all author names that contain the word 'collaboration' and all papers with more than 30 authors.

After this, we are left with a sample of 39,371 author IDs. From these, we randomly chose a subset of 28,000 as the 'training group'. The rest is our validation data by help of which we evaluate how well the neural network performs after training is completed. Note that this random split into training and validation data is done independently for each round of cross-validation.

The neural network

The neural network itself is built using Keras (Chollet et al. 2015) with the TensorFlow backend (Abadi et al. 2015).

We used a feedforward neural network, which means that the neural network consists of layers of neurons where the input of the neurons in one layer is the output of the neurons in the previous layer and the layers are not arranged as a cycle. The output of the first layer are the input data described in "Appendix 1", and the output of the last layer is taken as the output of the whole neural network. In our case, the last layer consists of ten neurons such that the neural network's output is list of ten real numbers.

For this network, the output of the neurons in one layer follows from the output of the neurons in the previous layer by

$$x' = \sigma(W \cdot x + b)\,. \tag{1}$$

Here, x' is a vector which contains the outputs of the N' neurons in one layer, x is a vector which contains the outputs of the N neurons in the previous layer, W is a real $N' \times N$ matrix whose elements are called the weights and b is a real vector with N' elements which are called the biases. Further, σ is a function which is applied to each element of the vector $W \cdot x + b$ and is called the activation function. The weights and biases are different for each layer, so that for each added layer one gets another weight matrix and another bias vector. These are the free parameters of the neural network which are determined by the training procedure.

During the training of the neural network, the output of the neural network is calculated with the training data described above as input and the weights and biases are optimized to get the output of the neural network for each author as close as possible to the actual $N_c(t_1, t_2)^{1/2}$. More precisely, the weights and biases are adjusted in order to minimize the so-called loss function which we take to be the mean squared error across all authors in the training set. Note that for a so-called fully-connected layer all elements of the weights matrix and the bias vector are independently adjusted, while for other types of layers, e.g. so-called convolutional layers, some structure is imposed. As will be explained below, we use both types of layers.

Since our dataset is not particularly large, we tried to avoid overfitting by reducing the number of parameters. This is achieved by the following network structure which has 12,750 parameters (see Fig. 1):

- The first layer is a convolutional layer and contains $k = 70$ neurons for each paper. Although convolutional layers are normally used to exploit translational symmetry in the input data, we use a convolutional layer with a convolution window of a single paper in order to ensure invariance under permutations of the papers. The effect of this

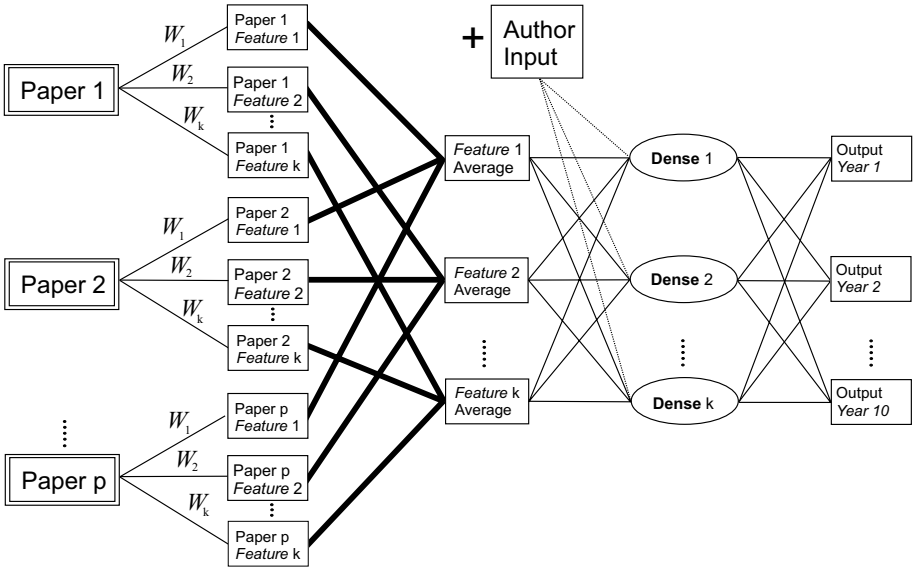

Fig. 1 Flow diagram of the neural network. Network elements surrounded by single thin lines are single neurons; those with double thin lines are collections of neurons. Connections shown with bold lines have a fixed weight of 1

is that each paper has a corresponding set of 70 neurons which see only the data from that paper, and each paper's neuron-set has matching weights and biases. The input to this layer contains every input except the broadness value, i.e., it contains all per paper input but not the per author input.

- In a next step, these 70 neurons per paper are reduced to 70 neurons in total by averaging each of the 70 neurons over the different papers. After this, no information about individual papers is left, and only average values are retained in the remaining 70 neurons. This layer does not add free parameters to the neural network. Note that the zero-padding makes this averaging equivalent to a summation with an author-independent normalization (see "Appendix 1").
- After that, a fully-connected layer with 70 neurons is added. In addition to the output of the previous layer, this layer obtains input which is specific to the author not to the individual papers.
- The final layer is a fully-connected layer with ten neurons with a ReLu activation function. The first neuron represents the prediction 1 year after the cutoff and the other neurons represent the differences between the prediction after n and $n-1$ years. For instance, if the neural network's output is $[5, 0, 1, \ldots]$, the corresponding prediction is 5 for 1 / 1 / 2009, 5 for 1 / 1 / 2010, 6 for 1 / 1 / 2011, etc. This ensures that the neural network's prediction is a monotonically increasing time series.

A detailed list of input data for each level can be found in "Appendix 1". The neural network architecture described above can be implemented in Keras with just a few lines of code which are reproduced in "Appendix 2".

The neurons, except for the neurons in the output layer, are taken to have tanh activation functions. Training is done using an Adam optimizer and a mean squared error loss

function. No regularization is employed and the final result is obtained after 150 epochs of training with a batch size of 50.

Training for 150 epochs takes about 15 minutes on a modern quad-core CPU with 3.8 GHz and 16 GB of RAM.

We would like to end this section with a comment on the way the neural network described above makes predictions for different years n after the cutoff. As described above, each year n corresponds to one of the 10 neurons in the very last layer of the neural network. An alternative would be to have 10 neural networks with only a single neuron in the output layer. Each of the networks would then be trained to make a prediction for one particular n. One might argue that this alternative leaves more freedom for the neural networks to learn the specific requirements in making a prediction for one specific n instead of for all n at the same time.

However, it seems this is not the case in practice, since we have tried both approaches with the only difference in the neural network architecture being the number of neurons in the output layer and the resulting performance was very similar.

One explanation for this could be that the cumulative h-index and $N_{\mathrm{c}}(t_1, t_2)^{1/2}$ are highly correlated for different n, so that making a prediction for a single year n is not much different from making a prediction for multiple years n. Another explanation could be that the first layers in the network learn representations which generalize across different years n, while the later layers use these representations to make the actual predictions depending on n.

Results

For both approaches, we will compare the neural network's performance to that of a naive h-index predictor which is given only the h-index of an author for which a prediction is to be made. By this naive h-index predictor we mean the following: for a given quantity to predict, e.g. the future cumulative h-index or $N_{\mathrm{c}}(t_1, t_2)^{1/2}$, take all authors in the training set with a given h-index h_0 at the time of the cutoff. Then, calculate the arithmetic mean of the quantity to be predicted and take this mean value as a prediction for authors in the validation set given their h_0.

Note that there may be authors in the validation set with, typically high, values of h_0 that are not present in the training set. For those authors, the prediction is determined as follows. First, a linear polynomial is fitted to the naive h-index predictor for the values of h_0 that can be calculated in the way described in the previous paragraph. Then, the value of the fitted polynomial is taken as the prediction for the other values of h_0.

We further compare the neural network's performance to that of a second, less naive, random forest baseline predictor (Breiman 2001). Like the neural network, this random forest predictor is trained on the training set and its performance is then evaluated on the validation set. We choose the hyperparameters of the random forest predictor to be the same as in Weihs and Etzioni (2017).

Comparison with the *h*-index

For the neural network trained to predict $N_{\mathrm{c}}(t_1, t_2)^{1/2}$, we are only interested in the prediction for $n = 10$ years after the cutoff. Therefore, we ignore the prediction of both the neural network and the naive h-index predictor for the first 9 years after the cutoff.

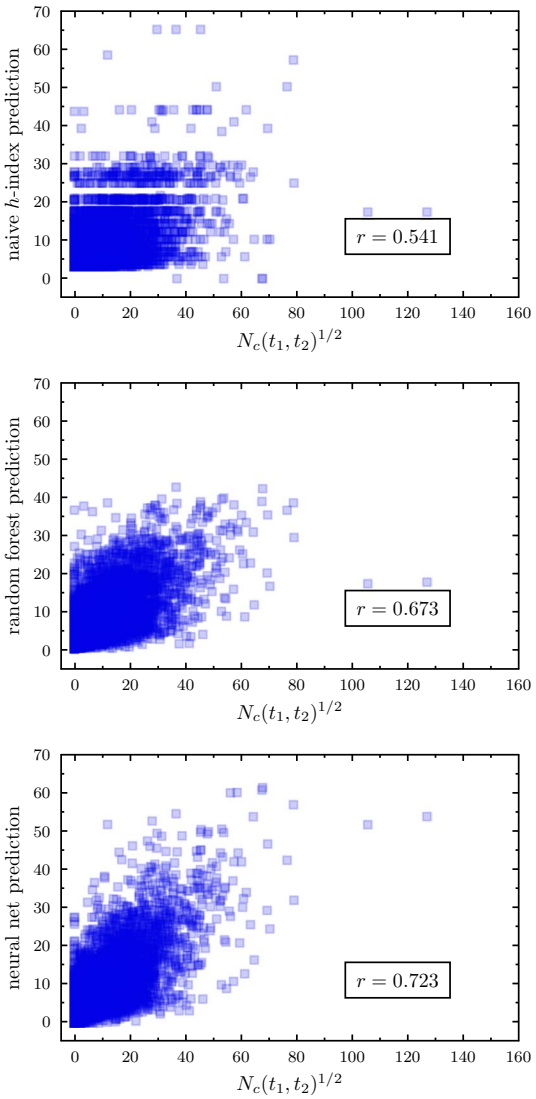

Fig. 2 The neural network's and the baseline predictors' predictions for $N_c(t_1, t_2)^{1/2}$ compared to the actual value for all authors in the validation dataset. Shown is the result from the first round of cross-validation

The correlation coefficients r betweeen the neural network's prediction and $N_c(t_1, t_2)^{1/2}$ is $r = 0.728 \pm 0.006$ (see Fig. 2). The error here and in the following is the standard deviation across the 20 rounds of cross-validation. In comparison, the naive h-index predictor described above yields $r = 0.552 \pm 0.009$ and the random forest predictor yields $r = 0.676 \pm 0.008$. We see that the neural network performs significantly better than the naive h-index predictor as well as the random forest predictor.

As one sees in Fig. 2, the network performs badly for authors with a very high number of citations. For those cases, the network's predictions are significantly lower than than the actual values. The likely reason for this is that the group of authors with a total number of citations larger than $(60)^2$ is very small (61 in total, about 18 in each validation set), so that

the network has no chance to learn how to properly predict them. We have removed this group of authors from the validation set to see how much this influences the network's performance, and find that the performance doesn't change by much.[1]

Next, we tested whether any one type of input data is especially important to the performance of the neural network by training the network with single types of input data separately removed. This, we found, barely changes the results. The biggest impact comes from removing the paper dates and the number of citations, which make the correlation coefficient drop to $r = 0.687 \pm 0.008$ and $r = 0.720 \pm 0.005$, respectively, which is still very close to the original $r = 0.728 \pm 0.006$. We may speculate that the network gathers its information from combinations of input which are themselves partly redundant, so that removing any single one has little effect.

Finally, we checked whether the neural net still performs better than the h-index when given only citation counts as input data. This resulted in a correlation coefficient of $r = 0.579 \pm 0.010$. We see that the neural net performs better than the h-index even with only the number of citations as input data. In contrast, the random forests predictor performs slightly better than the neural network with only the number of citations as input, $r = 0.582 \pm 0.009$, but this difference is not significant when compared to the error bars.

For more details on the comparison between our results and those of Hirsch (2007), please see "Appendix 3".

Comparison with earlier machine learning predictions

For the second approach, we compare the predicted with the actual cumulative h-index for $n = 1, 2 \ldots 10$ years after the cutoff date. We quantify the goodness of this prediction with the coefficient of determination, R^2, both for the neural network and for the naive h-index predictor discussed at the beginning of this section. Note that we calculate R^2 separately for each n by restricting the predictions as well as the actual values to that particular n. The result is shown in Fig. 3. The error bars show ± 1 standard deviations calculated from the 20 rounds of cross-validation.

We see that the neural net, the naive h-index predictor, and the random forest predictor are similarly predictive for $n = 1$, but the neural network's prediction becomes better in comparison for larger n. This agrees with the findings Acuna and Allesina (2012), Weihs and Etzioni (2017).

An alternative to R^2, also used in Weihs and Etzioni (2017), is the mean absolute percentage error (MAPE). We cannot calculate percentage errors for all authors in our data set since the set contains authors which have an h-index of 0. Therefore, at each n, we instead calculate the MAPE only for the set of authors with non-zero h-index. For $n = 10$ years, this set contains 571 authors. Since we trained the network for all predictors including authors with zero h-index, one must be careful with interpreting the MAPE calculated by excluding these authors. Nevertheless, these MAPE values for authors with non-zero h-index are shown in Fig. 4, left. We see that for $n \leq 2$, the

[1] More concretely, we have calculated the correlation coefficient and the mean absolute percentage error (MAPE, as discussed in the "Comparison with earlier machine learning predictions" section). The correlation coefficient drops by about 0.01 and the MAPE improves by about 0.1. Given that correlation coefficients cannot be simply compared between different subsets, the drop in the correlation cannot be interpreted straightforwardly. However, the small change in the MAPE indicates that the removed authors only have a small effect on the results.

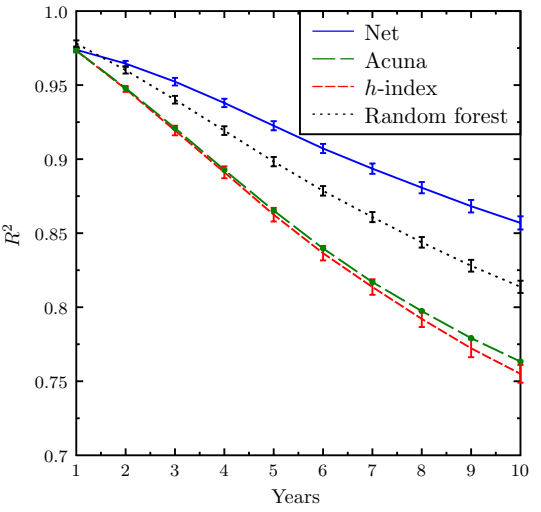

Fig. 3 R^2 of the prediction of the cumulative h-index as a function of years after cutoff

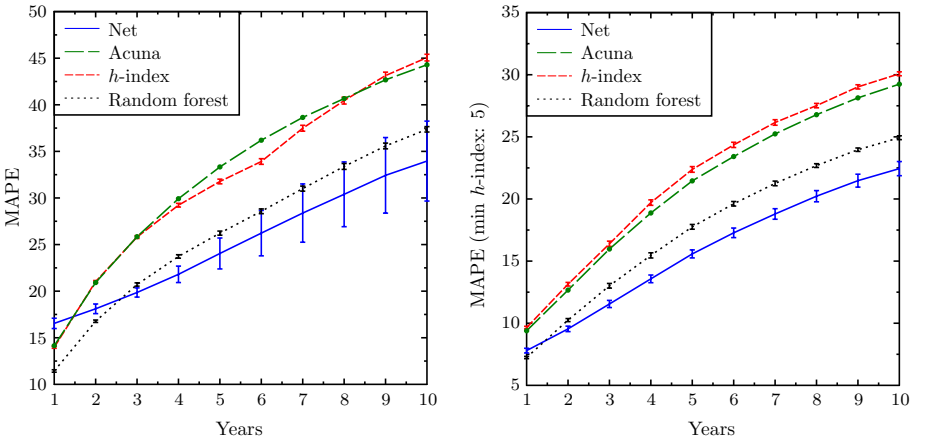

Fig. 4 Left: MAPE of the prediction of the cumulative h-index as a function of years after cutoff, only for authors with a non-zero true h-index at the respective n. Right: MAPE of the prediction of the cumulative h-index as a function of years after cutoff, only for authors with a minimum h-index of 5 at the respective n

baseline models have a better MAPE than the neural network. In contrast, for larger n, the neural network again performs better than the baseline models.

As one sees in Fig. 4, the fluctuations of the MAPE for the neural network are much larger than those of the other predictors, as indicated by the error bars. We think these large fluctuations in the MAPE are likely due to our choice of the loss function with which the neural network is trained. In particular, our loss function is the mean squared error which does not penalize large percentage errors as long as the absolute squared error stays small. For authors with small h-index, the network is therefore free to introduce relatively large percentage errors. This freedom can lead to large fluctuations of the MAPE, since the neural network can make this choice independently for each round

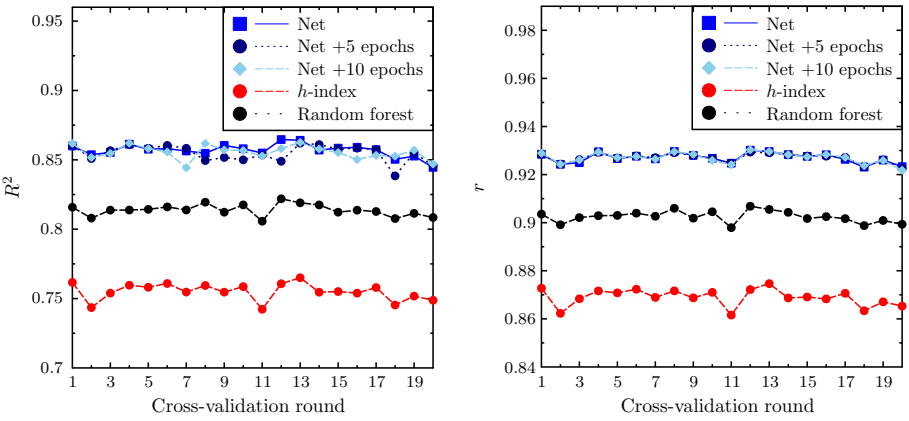

Fig. 5 Left: Neural network's and baseline predictors' R^2 after $n = 10$ years for each round of cross-validation. Right: Neural network's and baseline predictors' r after $n = 10$ years for each round of cross-validation

of cross-validation. Indeed, the mean squared error fluctuates much less than the MAPE (for $n = 10$, the mean squared error is 5.88 ± 0.15 compared to the MAPE's 34.0 ± 4.3).

To test whether authors with a small h-index are indeed the origin of the large fluctuations, we have repeated the calculation while excluding all authors with an h-index smaller than 5 at each n. As one sees in Fig. 4, right, this significantly reduces the errror bars for the neural network, but only slightly for the other predictors, as the above interpretation suggests.

As mentioned in the "Aim" section, there are fluctuations of the neural network's performance with the rounds of cross-validation. These fluctuations are illustrated in Fig. 5. In particular, Fig. 5 shows the neural network's R^2 (Fig. 5, left) and r (Fig. 5, right) after $n = 10$ years for the different rounds of cross-validation. We see that there are non-negligible fluctuations in the network's performance. A possible explanation could be that the fluctuations are due to the neural network overfitting on particular kinds of splits of the whole dataset into training and validation data. However, we think it is more likely that the fluctuations are due to intrinsic properties of the dataset. This is because the naive h-index predictor has only a few tens of parameters so that overfitting should not be an issue but Fig. 5 shows that there are comparable fluctuations in this naive h-index predictor's performance nonetheless.

In Fig. 6, left, we show the result for the neural network operating on the training and validation datasets of the first round of cross-validation when training for 5 and 10 additional epochs. We see that there are fluctuations of the neural network's performance with the training epoch which typically affect R^2 at or below the one percent level. In Fig. 6, right, we show the result of averaging across all round of cross-validation, where the averaging is done separately for the neural networks obtained after training for 150, 155, and 160 epochs. We see that the fluctuations with the training epoch have cancelled and the difference between the averages after training for 150, 155, and 160 epochs is negligible.

Quantitatively, our results give higher values of R^2 than both Acuna and Allesina (2012) (0.48 after 10 years) and Weihs and Etzioni (2017) (0.72 after 10 years). However, since not only our neural network but also our simple h-index predictor give higher R^2-values than Acuna and Allesina (2012) and Weihs and Etzioni (2017), this difference is probably partly due to the different datasets. We have therefore also applied the

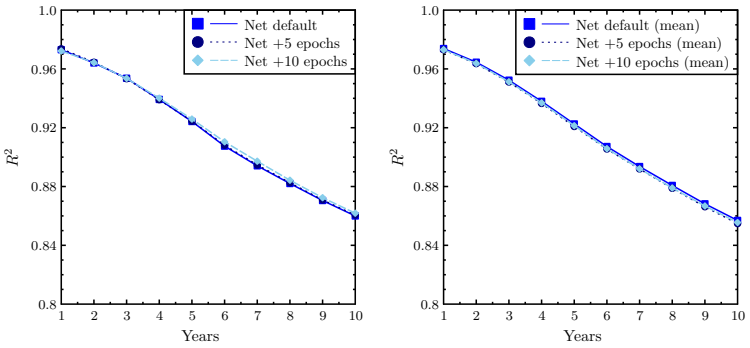

Fig. 6 Left: Network trained by our default value of 150 epochs, compared with training for 155 and 160 epochs for the first round of cross-validation. Right: Average across 20 rounds of cross-validation of the network trained by our default value of 150 epochs, compared with training for 155 and 160 epochs

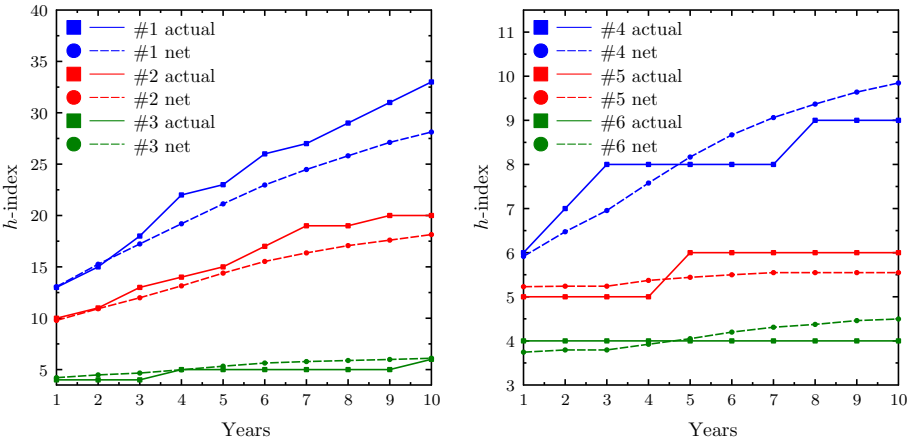

Fig. 7 Example trajectories for actual development of the h-index over time (solid/squares) and trajectories predicted by the network (dashed, circles) for the training and validation data of the first round of cross-validation

predictor proposed in Acuna and Allesina (2012) to our data-set, with the results shown in Fig. 3 (see "Appendix 4" for details). The predictability of our data-set is indeed higher than that studied in Acuna and Allesina (2012), but the prediction of our network still outperforms the previous study. Unfortunately, a similar direct comparison to the results from Weihs and Etzioni (2017) which used yet another data-set is not possible. Still, our value of $R^2 = 0.857 \pm 0.004$ after ten years is remarkably predictive, especially given that the methods we have employed here are likely to improve further in the soon future.

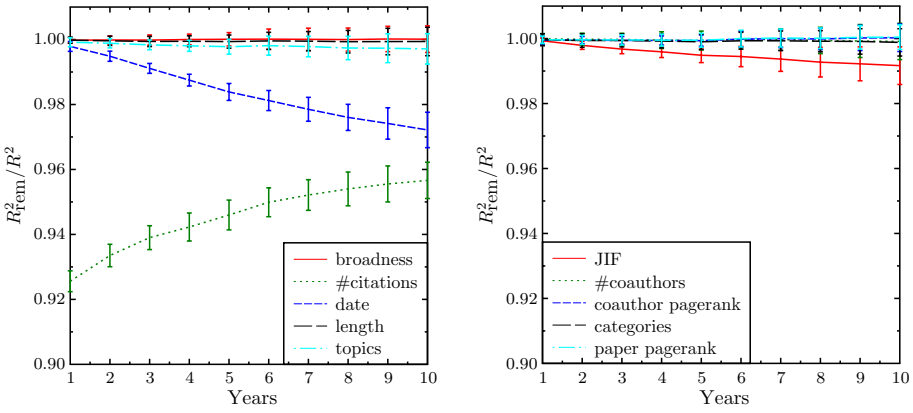

Fig. 8 Ratio of R^2 of the neural net to the same indicator for the neural net with various input data removed, $R^2_{\rm rem}$, as a function of years past cutoff

Since the value of R^2 by itself is not so illuminating, we show in Fig. 7 some examples for which we display the actual h-index versus the network-prediction with the training and validation datasets from the first round of cross-validation.[2]

Here too we have investigated how important various input data are to the neural network's performance by removing one at a time. The results are shown in Fig. 8, where we plot the ratio of the coefficients of determination with all input (R^2) and with certain inputs removed ($R^2_{\rm rem}$). The lower the ratio, the more important the removed input was. The error bars show ± 1 standard deviations calculated from the 20 rounds of cross-validation.

We see that for $n = 1$, the number of citations is the only important input, while for $n > 1$ other inputs gain importance with the number of citations still being the most important one. That the citation data are most important for $n = 1$ agrees with the results of Acuna and Allesina (2012) and Weihs and Etzioni (2017). We also see from Fig. 8 that $R^2_{\rm rem}/R^2$ is always larger than 0.9 which again indicates that the input data are partly redundant. However, the changes we notice due to some input removals are so small that fluctuations in the training results are no longer negligible.

We have investigated further how important various input data are by removing all except one at a time to give us an idea how much information can be extracted from single types of input. The results are shown in Fig. 9 where we plot the ratio of the coefficients of determination with all input (R^2) and with all except certain inputs removed ($R^2_{\rm rem,except}$). We see that also with only a single input at a time the neural network's $R^2_{\rm rem,except}/R^2$ never drops below 0.6. The single input from which the neural network can get the best performance is the number of citations, which gives $R^2_{\rm rem,except}/R^2 > 0.9$ for all n. But also the JIF comes close to an $R^2_{\rm rem,except}/R^2$ of 0.9. Note that there are inputs like the paper pagerank which give a relatively high $R^2_{\rm rem,except}/R^2$ but a $R^2_{\rm rem}/R^2$ close to 1. This again indicates that our inputs are partly redundant.

When interpreting these numbers, however, one must keep in mind that the neural network always implicitly knows the number of papers of an author in addition to the

[2] The reader be warned that these examples were not randomly chosen. We hand-selected a set of noticeably different h-index outcomes for purely illustrative purposes.

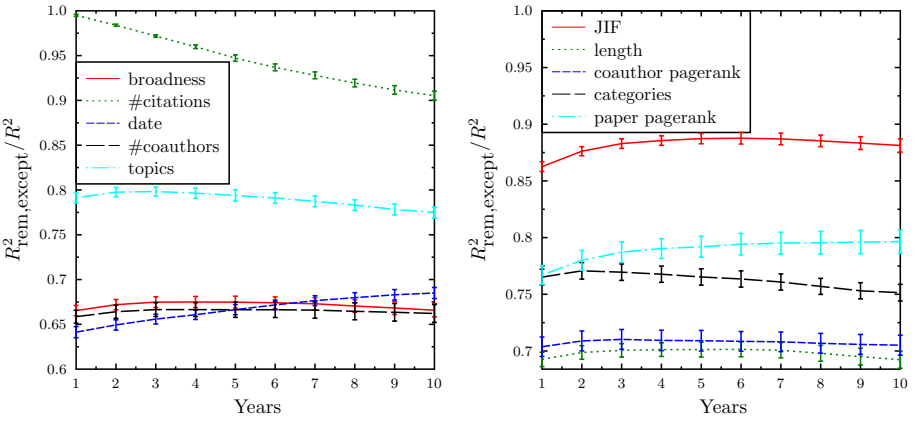

Fig. 9 Ratio of R^2 of the neural net to the same indicator for the neural net with all except certain input data removed, $R^2_{\text{rem,except}}$, as a function of years past cutoff

input we give it explicitly, see the network architecture described in the "The neural network" section and "Appendix 1".

Discussion

We have demonstrated here that neural nets are powerful tools to make predictions for the citations a researcher's work accumulates. These predictions are likely to improve in the future. One of the major limitations of this present study, for example, is that our sample does not include papers which are not on the arXiv, and that about one of five published papers could not be associated with a Journal Impact Factor (see "Appendix 1"). But the more bibliometric data becomes available the more input the network can be fed, and thus predictivity is bound to become better for some more time.

The methods we used here are straight-forward to implement and do not require much computing resources. It is thus foreseeable that in the near future the use of neural nets to predict researcher's promise will become more widely spread.

We therefore want to urge the community to not ignore this trend in the hope it will go away. It would benefit academic research if scientists themselves proposed a variety of predictors, and offered a variety of data, to more accurately present the variety of ways to do high-quality research.

In this work we focused on the *h*-index in order to compare our results with previous results, and also because this value is easy to extract from existing data. But a lot of valuable information about researchers and their work is presently difficult or impossible to obtain and analyze. For example, how often researchers are named in acknowledgements, their seminar and conference activity, the frequency by which they act as peer reviewers and/or editors, or how specialized their research topic is. All these are important factuals about the many individual approaches to research. Providing and analyzing such data would enable us to develop measures for success tailored to specific purposes and thereby avoid that researchers focus efforts on optimizing citation counts.

Acknowledgements Tobias Mistele thanks Sebastian Weichwald for helpful discussions. This work was supported by the Foundational Questions Institute (FQXi).

Appendix 1

The building blocks of most of the input data for the neural network are lists where the position in the list corresponds to a paper and the value at a certain position corresponds to some input data associated with the corresponding paper. E.g. there is a list containing the number of citations for each paper of the given author which in Python syntax would look like [130, 57, …], meaning the most-cited paper of this author has 130 citations, the second-most cited paper has 57 citations etc. A corresponding list for the number of coauthors would look like [0, 1, …], meaning the paper with 130 citations was a single-authored paper, the paper with 57 citations has two authors etc.

Since different authors have different numbers of papers, these lists will have different lengths for different authors. However, our neural network requires a fixed-size input. Therefore, we take the lists for all authors to have the same length as those of the author with the largest number of papers, namely 169. The positions in the lists that do not correspond to a paper are filled with zeros.

Note that the averaging in the second layer of our neural network (see the "The neural network" section) includes the neurons that do not correspond to a paper. Therefore, this averaging is equivalent to a summation normalized by an author-independent constant, namely the number of papers of the author with the largest number of papers. Since the neural network knows which neurons correspond to a paper (see below), it can, in principle, average over the number of papers of each individual author in a later layer. However, we do not impose this in the network architecture. This is because it makes sense for features to contribute cumulatively, since authors with more papers are likely to receive more citations.

The biggest part of the input to the neural network is then the list of all the lists described above and consists of

1. A list which contains 1 at each position in the list which corresponds to a paper and 0 at each position which corresponds to zero-padding.
2. A list which contains the number of citations of each paper.
3. A list which contains the publication date of each paper relative to the cutoff date.
4. A list which contains each paper's pagerank (Samuel 2015), an interactive measure of relevance that works similar to Google's pagerank algorithm just that, instead being based on hyperlinks, a paper's pagerank is based on the citation graph. The pagerank is calculated from the citation graph at the cutoff date, 1/1/2008.
5. A list which contains each paper's length.
6. A list which contains 0 for each paper with an empty journal reference and 1 for each paper with a non-empty journal reference.
7. A list which contains the JIF of the journal each paper is published in (further details below). If a paper is not published or no JIF is known for a journal, we take the corresponding input to be 0. The JIFs are taken at the cutoff date.
8. A list which contains the number of coauthors of each paper.
9. Three lists which contain the coauthors' minimum, maximum, and average pagerank. Here, the pagerank is calculated from the coauthor graph at the cutoff date.

10. For each arXiv category a list which contains zeros except at position which correspond to papers which are in the respective category. In order to reduce the amount of data, categories of the form a.b are all treated as category a, e.g. astro-ph.CO and astro-ph.HE are treated as the same category.
11. For each paper, a 50-dimensional vector representing a paper's latent topic distribution, obtained from the keyword analysis done by Price and Hossenfelder (2018) when operating on the arXiv data up to the cutoff.

The final input to the neural network is a 'broadness' value calculated from a keyword analysis (Price and Hossenfelder 2018) with the same data as the paper topics described above. This broadness quantifies how widely spread the topics which an author publishes on are over all arXiv categories. Since this broadness value is one value per author and not one value per paper it is handled separately from the other inputs to the neural network.

All input data except the categories and the paper vectors is normalized to unit variance and zero mean. More specifically, for each input (e.g. number of citations, publication date, broadness, ...) a transformation is determined from the training data such that this transformation brings the given input to zero mean and unit variance for the training data across all authors. This transformation is then applied both to the training and the validation input data.

To assign the JIFs, we associate papers in our database with a journal by heuristically matching the journal reference given in the arXiv metadata to the journal abbreviation from Clarivate Analytics (2017). Concretely, we reduce both the journal reference in the arXiv metadata and the journal abbreviation from Clarivate Analytics (2017) to lower-case alphanumeric characters and cut at the first numeric character. Next, we remove the suffixes 'vol' and 'volume' if present. If the two values obtained this way are identical, we consider the given arXiv paper to be published in the corresponding journal.

To reduce the number of papers where this procedure does not work, we have further used a manually assembled translation table. This table contains identifications between reduced arXiv journal references to journals from Clarivate Analytics (2017) for which the method outlined in the previous paragraph does not work. The table allows us to match the 69 most frequent reduced journal references from the arXiv that could not be mapped by the previous method. By this procedure, we have assigned a Journal Impact Factor to 378,134 of the 477,176 papers with a non-empty arXiv journal reference.

Appendix 2

This is how the neural network architecture described in the "The neural network" section can be implemented with Keras (Chollet et al. 2015). The full code for reproducing our results can be found at https://github.com/tmistele/predicting-citation-counts-net.

```
from keras.models import Model
from keras.layers import Input, Dense, Conv1D, concatenate, \
    GlobalAveragePooling1D

perpaper_inputs = Input(shape=perpaper_shape,
                        name='perpaper_inputs')
perauthor_inputs = Input(shape=perauthor_shape,
                         name='perauthor_inputs')

tmp = Conv1D(
    filters=70,
    kernel_size=1,
    strides=1,
    activation=activation,
    input_shape=perpaper_shape)(perpaper_inputs)

tmp = GlobalAveragePooling1D()(tmp)

tmp = concatenate([tmp, perauthor_inputs])
tmp = Dense(units=70, activation=activation)(tmp)

outputs = Dense(units=10, activation='relu')(tmp)

model = Model(inputs=[perpaper_inputs, perauthor_inputs],
              outputs=outputs)
```

Appendix 3

The correlation coefficient obtained from the naive *h*-index predictor is roughly the same as the correlation coefficient obtained by plotting $N_c(t_1, t_2)^{1/2}$ over the *h*-index at the time of the cutoff for both the training and validation data and calculating the correlation coefficient from that, see Fig. 10. Note that this second way of calculating a correlation coefficient corresponds to what was done in Hirsch (2007).

The correlation coefficients from Fig. 10 are consistenly smaller than those of the sample PRB80 from Hirsch (2007) but higher than those of the sample APS95 from Hirsch (2007). See Hirsch (2007) for a discussion of the differences between the samples PRB80 and APS95 regarding their differing correlation coefficients. Our sample differs from both PRB80 and APS95 in both the data source and the cuts applied. Therefore, it is not surprising that there are differences in the resulting correlation coefficients and the results are not directly comparable. One important difference is that we employ the same cutoff, 1/1/2008, for all authors while Hirsch (2007) applies a different cutoff for each author at 12 years after each author's first paper.

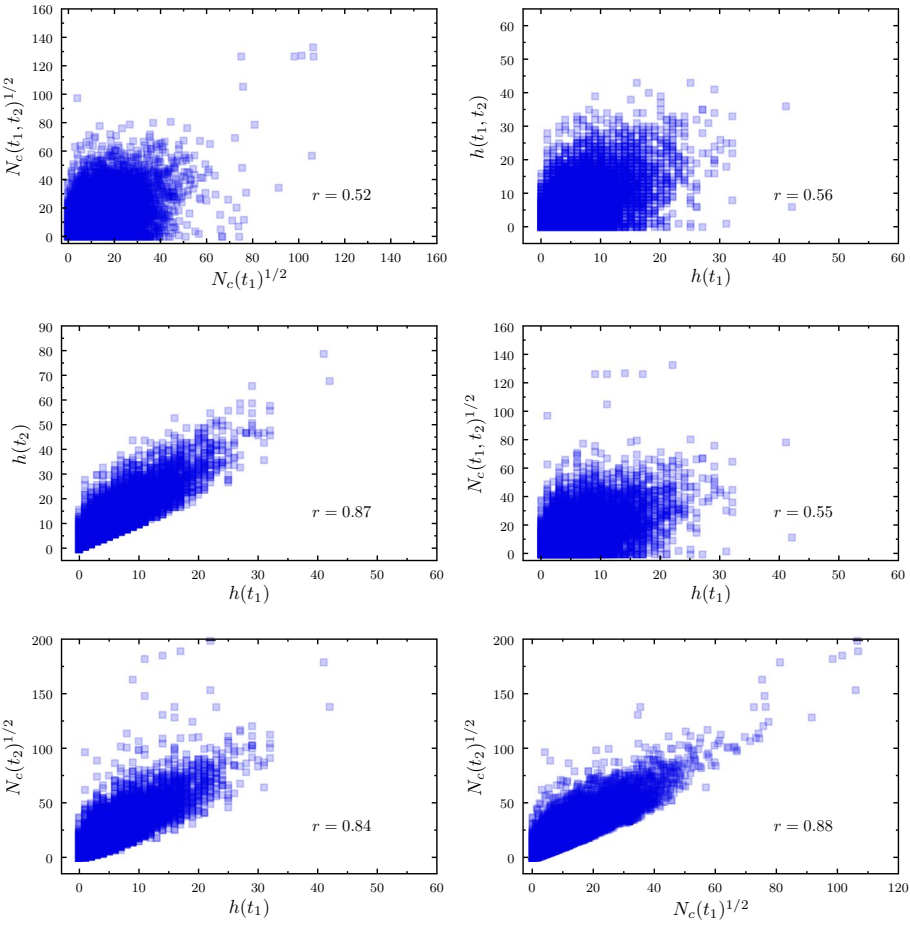

Fig. 10 Correlation of various quantities calculated from the complete dataset including both training and validation data with the respective correlation coefficient r. The notation is that of Hirsch (2007). E.g., $h(t_1)$ is the h-index as calculated from an author's first 10 years of publishing, $h(t_2)$ is the cumulative h-index after an author's first 20 years of publishing, and $h(t_1, t_2)$ is the h-index calculated from the second 10 years of an author's publishing excluding papers written and citations received outside this period of time. For different datasets, these plots can be found in Hirsch (2007)

Appendix 4

We used what the authors of Acuna and Allesina (2012) refer to as the "simplified model," that—as they have shown—performs almost as well as their full model on their data-set. We changed the selected journals from Nature, Science, Nature Neuroscience, PNAS and Neuron to Science, Nature, PNAS, and PRL. As laid out in the supplementary material of Acuna and Allesina (2012), we used the R-package 'glmnet' with $\alpha = 0.2$. Note that we did not employ our own Monte Carlo cross-validation here. Instead—as was done in Acuna and Allesina (2012)—we relied on the cross-validation included in the 'glmnet' package.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., & Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/. Accessed Apr 2018.

Acuna, D. E., & Allesina, S. (2012). Predicting scientific success. *Nature*, *489*(7415), 201–202.

Alonso, S., Cabrerizo, F. J., Herrera-Viedma, E., & Herrera, F. (2009). *h*-index: A review focused in its variants, computation and standardization for different scientific fields. *Journal of Informetrics*, *3*(4), 273–289.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

Chollet, F., et al. (2015) Keras. https://github.com/keras-team/keras. Accessed Apr 2018.

Clarivate Analytics. 2001–2009 journal citation reports (2017).

Csiszar, A. (2017). The catalogue that made metrics, and changed science. *Nature*, *551*(7679), 163–165.

Dong, Y., Johnson, R. A., & Chawla, N. V. (2014). Will this paper increase your *h*-index? Scientific impact prediction. *CoRR*, arXiv:1412.4754.

George, D. P., & Knegjens, R. Paperscape. http://paperscape.org.

Hirsch, J. E. (2007). Does the *h* index have predictive power? *Proceedings of the National Academy of Sciences*, *104*(49), 19193–19198.

Nezhadbiglari, M., Gonçalves, M., & Almeida, J. M. (2016). Early prediction of scholar popularity. *2016 IEEE/ACM joint conference on digital libraries (JCDL)* (pp. 181–190).

Open archive initiative, arxiv download URL. https://arxiv.org/help/oa/index. Accessed Apr 2018.

Paperscape documentation on github. https://github.com/paperscape/paperscape-data/. Accessed Apr 2018.

Price, T., & Hossenfelder, S. (2018). Measuring scientific broadness. arXiv:1805.04647 [physics.soc-ph].

Samuel, M.H. (2015) Pagerank + sparse matrices+ python (ipython notebook). http://blog.samuelmh.com/2015/02/pagerank-sparse-matrices-python-ipython.html. Accessed Dec 2017.

Van Noorden, R. (2012). Metrics: A profusion of measures. *Nature*, *465*, 864–866.

Waltman, L. (2015). A review of the literature on citation impact indicators. arXiv:1507.02099 [cs.DL].

Weihs, L., & Etzioni, O. (2017). Learning to predict citation-based impact measures. In: *2017 ACM/IEEE joint conference on digital libraries (JCDL)* (pp. 1–10).

Yan, R., Huang, C., Tang, J., Zhang, Y., & Li, X. (2012). To better stand on the shoulder of giants. In *JCDL*.

Zhang, C., Liu, C., Yu, L., Zhang, Z.-K., & Zhou, Tao. (2016). Identifying the academic rising stars. *CoRR*, arXiv:1606.05752.