Final Report : CSE 519 Data Science

Ranking Academic Research Papers

Abstract: Ranking metrics to evaluate academic papers and researchers are important because these are used by funding bodies and employers to allocate grants and jobs. Ranking scientific papers helps researchers to find related work of high quality. In this paper, we build a ranking metric that applies the principles of data science to evaluate academic papers and researchers and compare it with the standard ranking metric, h-index.

Introduction: This is a well studied problem and several metrics have been defined that rank academic papers and journals based on citation counts, time of publication, even domain and its reach. The most popular, namely h-index, developed by Jorge E. Hirsch, is an author level metric that measures the citation impact of the publication of a researcher. It is defined as the highest number of publications of a scientist that received h or more citations each. A lot of attempts have been made to get a better alternative for h-index [1,2,3], one that can cover a broader perspective : m-index, g-index, c-index etc. We use ideas from them and develop a new ranking metric that can consider more features (like citation sentiment analysis, pagerank score, topic modelling, time analysis and network reach) and give more accurate predictions. In the remainder of the paper, we describe which dataset we use, how we preprocessed the data and how we develop our ranking metric based on the parameters extracted from the dataset.
Using our ranking metric we perform the following given tasks:

1) Identify top 100 ranked researchers from multiple disciplines and see how it stacks up against their respective h-index.
2) Using a citation network graph, devise a “reach” function which can identify the degree of a paper’s influence.
3) Identify examples of researchers with substantial differences between our metric and h-index. Develop an evaluation to decide who is better in these cases.
4) Do a time analysis of paper citation count against our metric.
5) Identify papers on Arxiv which should become popular or important.

Primary Dataset:
Citation Network Dataset [4] is our primary dataset.
It contains citation data extracted from DBLP, ACM, MAG (Microsoft Academic Graph), and other sources. It contains following attributes for each paper:

#* --- paperTitle
#@ --- Authors
#t ---- Year
#c --- publication venue
#index 00---- index id of this paper
#% ---- the id of references of this paper (there are multiple lines, with each indicating a reference)
#! --- Abstract

Our primary dataset contains two different types of data. One json formatted and other text data starting with a specific prefix.
The existing set of features in our Database included 8 total data columns:

abstract	non-null object
authors	non-null object
id	non-null object
n_citation	non-null int64
references	non-null object
title	non-null object
venue	non-null object
year	non-null int64

There were a total of 3079007 papers in the dataset. The two columns (abstract, references) have some null/missing values. Rest 6 columns don't have any null values.

Data Collection:(Further cleaning and preprocessing explained below) For the text data, we wrote python script to parse it and convert into CSV (comma-separated) format. Then read CSV file into DataFrame. For the json data, data was divided into 4 parts. We loaded data into 4 DataFrames and then merged complete data into single Dataframe.

External Datasets:
We used below datasets to enhance our primary dataset and hence make better predictions

1. Google Scholar DataSet :
   This is a huge dataset of scholarly literature. Among other things, it maintains a profile of each author that comprises of a list of titles of all the author's publications, affiliation, total citation count, h-Index, i10-index, list of co-authors and some other attributes. Google Scholar doesn't provide an API to get the data. Hence we used scholarly [5] which is a python library to extract author information.

   Below is sample api call and response that we used:

```
import scholarly
print(next(scholarly.search_author('Steven Skiena')))
{
      '_filled': False,
```

```
'affiliation': 'ProfessorofComputerScience,
StonyBrookUniversity',
'citedby': 12979,
'email': '@cs.stonybrook.edu',
'id': 'fnE2dSoAAAAJ',
'interests': ['Algorithms',
'DataScience',
'ComputationalBiology'],
'name': 'StevenSkiena',
'url_picture':
'/citations?view_op=small_photo&user=fnE2dSoAAAAJ&citpid=2'
}
```


This dataset helped in extraction of features like domain of the author, reach of the author, validating the h-index etc.

Ways in which we used the above dataset:

1) Presuming that 'interests' is same as the author's major research area, we gave more weightage to those citations which are outside the author's research area compared to citations from same area in our metric.

2) While creating the collaboration graph of an author, we used the 'affiliation' attribute. We weighed citation from same affiliation low compared to one not from affiliation.

2. Eigenfactor scores dataset :

This dataset contains the eigenfactor scores of roughly 11k journals which they update periodically. The eigenfactor score reflects the impact of a journal where a higher score means bigger impact all else being equal.

Since there is no direct API available, we used below way to fetch the eigen scores of journals:

Way : By scraping from their web page:[6]

We used this while weight the impact of the venue for a given paper. We made sure that the papers that are published in venues with high eigen-vector get more venue score.

3. Scholarly Publishing – MIT Libraries

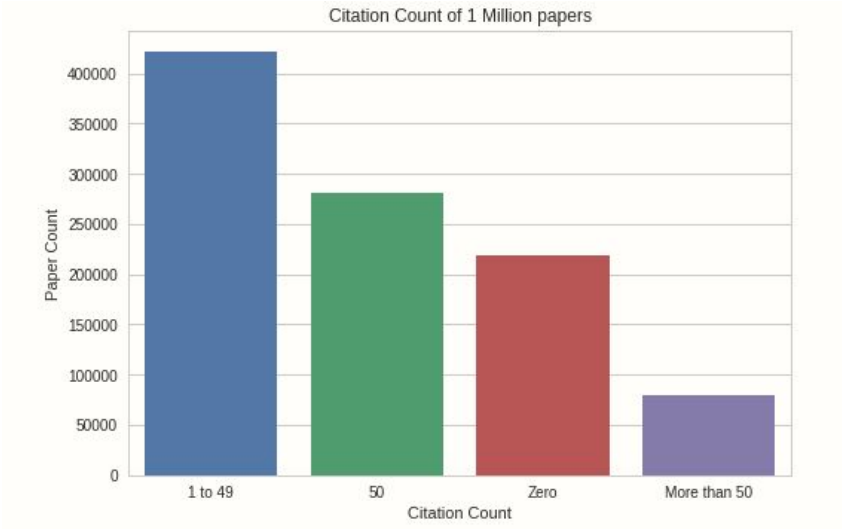
This data set contains a bunch of scholarly resources (eg:IEEE Xplore API, arXiv API) that make their APIs available for use.

We explored this although could not use it to expand our dataset. Source:[7]

Data Preprocessing:

For further cleaning and preprocessing the data, we performed the following steps:

1) For null abstract values, we imputed with an empty string
2) For null references, we imputed it with an empty list
3) Distribution of Citation Count vs Paper Count :
    a) Over **42%** of the papers in the dataset have citation count between 1 to 49
    b) An interesting observation is that around **28%** of papers have citation count exactly equal to 50
    c) Around **22%** of papers don't have any citation count
    d) Around **8%** of papers have more than 50 citations



4) Distribution of Paper Count over year :
    a) In this dataset, we have papers roughly between the year 1930 and 2020.
    b) Most of the papers are of recent years (post 2000). Perhaps a major portion of the dataset was generated from sources where old publications are not recorded.

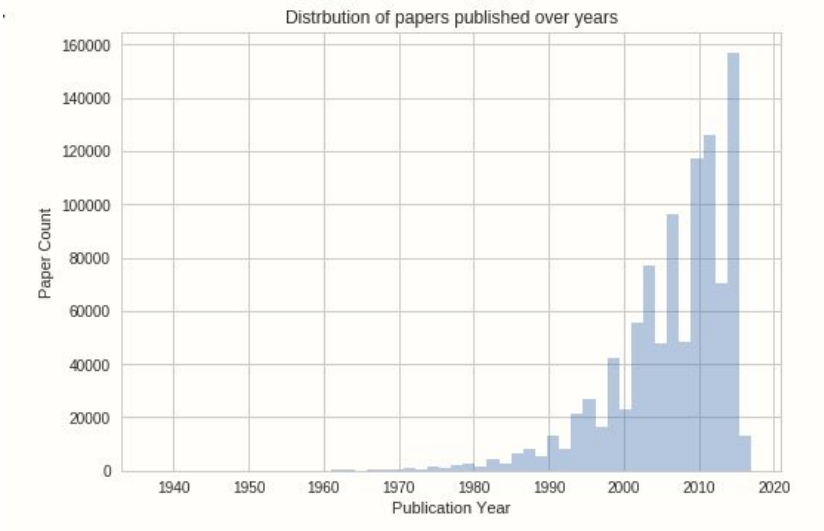


Figure 2 : Distribution of paper count over year

Features extracted from the Dataset:
We extracted the following features from the dataset to model our Ranking metric:

1) Domain Extraction (Topic Modelling using LDA) [8]
    a) Our dataset contained two fields namely ‘abstract’ and ‘title’ of the paper, both of which could be used to extract information about what domain the paper belongs.
    b) We used the title of the paper for the same because any title best represents the class of any object in few words.
    c) We used a topic modelling algorithm in NLP, called LDA for the same which is basically a type of unsupervised learning. The procedure is explained below:
    d) Few preprocessing steps were needed like Tokenization (split by space), Stopwords removal (words like a,an,the), lemmatization (3rd person to 1st, Past verb to present) and stemming(root forms of verbs).
    e) We used gensim and nltk libraries for the same.
    f) Then we created a bag of words for the titles of all the papers which we store in a dictionary containing how many words and how many times those words appear in that title.
    g) We used this bag of words to run our LDA model, giving the number of topics as 10. The output of our LDA model is as shown below:

Topic: 0	Word: 0.023*"network" + 0.014*"wireless" + 0.011*"sensor" + 0.008*"time" + 0.008*"rout" + 0.008*"system" + 0.007*"control" + 0.007*"power" + 0.007*"base" + 0.007*"perform"
Topic: 1	Word: 0.008*"recognit" + 0.008*"speech" + 0.007*"base" + 0.007*"learn" + 0.007*"model" + 0.006*"text" + 0.005*"machin" + 0.005*"classif" + 0.005*"featur" + 0.004*"analysi"
Topic: 2	Word: 0.005*"polygon" + 0.005*"digit" + 0.004*"expert" + 0.004*"cmos" + 0.004*"librari" + 0.004*"squar" + 0.004*"intersect" + 0.004*"technic" + 0.004*"span" + 0.004*"topic"
Topic: 3	Word: 0.016*"graph" + 0.010*"problem" + 0.009*"algorithm" + 0.009*"program" + 0.008*"logic" + 0.007*"tree" + 0.007*"bound" + 0.006*"approxim" + 0.006*"complex" + 0.006*"linear"
Topic: 4	Word: 0.019*"imag" + 0.010*"base" + 0.009*"object" + 0.008*"segment" + 0.008*"video" + 0.007*"detect" + 0.007*"motion" + 0.007*"track" + 0.007*"recognit" + 0.006*"model"
Topic: 5	Word: 0.007*"swarm" + 0.006*"particl" + 0.006*"net" + 0.006*"gene" + 0.005*"petri" + 0.005*"model" + 0.005*"base" + 0.005*"optim" + 0.005*"data" + 0.004*"ident"
Topic: 6	Word: 0.008*"algorithm" + 0.008*"peer" + 0.007*"markov" + 0.006*"optim" + 0.006*"model" + 0.006*"base" + 0.005*"method" + 0.005*"hide" + 0.005*"fuzzi" + 0.005*"cluster"
Topic: 7	Word: 0.006*"linear" + 0.006*"algorithm" + 0.006*"nonlinear" + 0.006*"method" + 0.005*"problem" + 0.005*"filter" + 0.005*"matrix" + 0.005*"model" + 0.005*"estim" + 0.005*"analysi"
Topic: 8	Word: 0.012*"mobil" + 0.012*"servic" + 0.009*"environ" + 0.008*"base" + 0.008*"manag" + 0.008*"agent" + 0.008*"network" + 0.007*"system" + 0.007*"robot" + 0.007*"virtual"
Topic: 9	Word: 0.011*"model" + 0.010*"base" + 0.009*"inform" + 0.008*"system" + 0.008*"knowledg" + 0.008*"softwar" + 0.008*"process" + 0.007*"data" + 0.007*"approach" + 0.006*"design"

Figure 3 : List of Topics identified by LDA model

    h) Basically, our model clusters the titles into 10 separate topics where clusters like [polygon, digit, square etc] represent papers related to mathematics domain, clusters like [recognition,text,learn,machine etc] represent papers in Machine learning domain and so on.

i) Our model doesn't directly give us the domain of a paper but groups similar titles together and gives 10 such clusters. However it also gives a list of top words corresponding to each cluster and we use this information to infer the domain. For example, one of our cluster is represented as [image,base,object..] sorted in descending order of most similar words. Hence it is safe to assume that most of these paper titles would be similar to image and hence their work would be in image domain (image classification, computer vision etc)

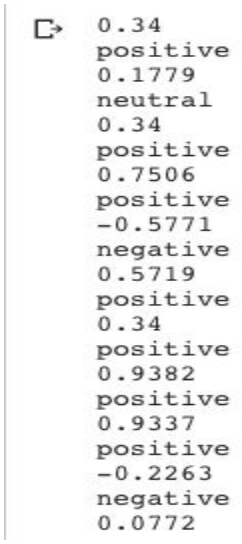
j) After adding the domain feature to our dataset (calculating the domain for each row of our training data), we use label encoding to encode these domains to feed it to our prediction model.

2) Citation sentiment analysis**:** [9]

The intuition behind the metric is that if a paper is cited many times but most of the times it is being criticized in those citations then more citations might actually mean bad rather than good for the paper/author. And our metric should consider this while ranking.

The way we figure out this is by doing a sentiment analysis on the abstract of the paper and getting a good or a bad score (eg. +1 or -1 or 0) depending on whether the tone was positive or negative or neutral. We performed Sentiment analysis using sentiment Intensity analyser from nltk library's sentiment.vader. There were some abstracts that were null and we dropped them since there was no way to calculate sentiment out of them. Then, We calculated the polarity score of each abstract and if it was greater than 0.2, then we assumed the sentiment of the abstract to be positive and gave it a score of +1. If the polarity score came out to be less than -0.2, then we assumed the sentiment of the abstract to be negative and gave a score of -1. If it was between (-0.2,0.2), then we assumed the sentiment to be neutral and gave it a score of 0. This sentiment score (-1,0,1) implied paper tone, about that topic, was positive, negative or neutral.

For eg. these were the sentiments of the top 10 papers in our dataset (along with the polarity scores):



```
0.34
positive
0.1779
neutral
0.34
positive
0.7506
positive
-0.5771
negative
0.5719
positive
0.34
positive
0.9382
positive
0.9337
positive
-0.2263
negative
0.0772
```

And here is the total number of positive, negative and neutral sentiments in a subset of our database. As we can see, majority of papers have positive sentiments which is natural since most papers will always talk in the favor of their titles.

```
[ ]  df['sentiment'].value_counts()

 1    258079
-1     53956
 0     39979
Name: sentiment, dtype: int64
```

Figure 5 : Distribution of sentiment types in our dataset

Now, Using this sentiment score for each paper, we calculated the citation sentiment score in this way: Suppose a paper is cited by n papers, then the total citation sentiment score would be the sentiment score (calculated earlier) of each of the paper. This comes from the intuition that if suppose, a paper cites some other paper but its speaking tone is negative then probably it does not have good things to say about the cited paper. (This is just a basic approach. It definitely can be improved given the content of the paper.) And hence the citation impact should actually be negative.
Consider an example,
Say a paper P is cited by P1,P2 and P3. And sentiment score of P1 is 1, P2 is -1, and P3 is 1. Then the total citation sentiment score of P would be 1-1+1=1. Which is positive which means that the citations had a positive impact overall.
We use this citation sentiment score in building our ranking metric based on the intuition that more the positive citation score, more the Rank of the paper.
Finally, in this dataframe we show the citation sentiment score of the top 10 papers according to our ranking metric:

id	sentiment	citation_sentiment	ranking_score	Rank
6a6b9aa6-683f-4c7c-b06e-9c3018d10fd3	-1	1613.0	3661.244512	1.0
3fb43b00-905c-4a08-934d-198ea4eb66c3	1	486.0	1895.095585	2.0
a662a4e7-415e-417e-8a8f-fe085d7e487f	1	195.0	1814.607493	3.0
9d912297-e52f-4ab6-add4-633e0f263933	1	830.0	1307.541050	4.0
3023929a-c93e-49c5-b03f-7fb0414d94df	1	535.0	1021.236758	5.0
b68fc787-7817-421e-8e66-8a98ab9db1ad	1	390.0	992.011549	6.0
96ff09f7-e819-4de9-aaf8-9eac2f5fa751	0	317.0	928.630043	7.0
32ba673c-4d72-41a9-8596-272a9009cbb5	-1	1.0	923.543508	8.0
51f1493d-ce3b-4589-8373-55940026fecd	1	266.0	913.959390	9.0
c061069f-29d1-46d4-9974-dede8d5461f9	-1	563.0	865.399161	10.0

Figure 6 : Citation Sentiment Score of top 10 papers as per our ranking metric

As we can see, the papers which have high Rank(according to our metric) usually also have high citation sentiment score. However there can be some exceptions (see rank 8), which is due to the other factors that we consider in addition to sentiment.

3) Time factor score (used in Time analysis):
This is a feature to consider a fact that in any time frame, because of some technology boom, more papers got published. For example, after 2012, in NLP domain, neural networks emerged as a powerful machine learning tool which leads to many publication in this domain.
Time factor score = ('n_citation')/('year_citation_sum' * 'year_citation_max')
   a. year_citation_count = Calculated total papers published in a year
   b. year_citation_sum = Sum of citation count of all papers published in a year
   c. 'n_citation' = Citation count of a paper

4) PageRank:
Page rank is very effective algorithm in predicting rank of the paper [10]. Using Pagerank algorithm, we developed weighted pagerank value based on citation network of papers. This page rank value of a paper will be high if it is cited by many other high-impact (high page rank value) papers. PageRank value of the paper could be more, even if has less number of citations, but has citation from quality papers. Thus those quality papers will increase quality of this paper.

We used following equation to calculate the pagerank value of the paper,

$$PR(p_i) = (1-d)/N + d*\sum (PR(p_j) / L(p_j))$$

where $p_1, p_2, p_3, \ldots p_n$ are the papers (nodes) in the network and N is the total number of papers. Summation is over all the papers p_j, which cited paper p_i. d is the damping factor which was set to 0.15. $L(p_j)$ is defined as the number of papers cited by p_j.

For pagerank calculation, we assumed initial value of pagerank of all papers then iteratively updated those values. Below is the algorithm for pagerank calculation:
1. Initialized pagerank value of all the papers by value = 100.
2. References data in the dataset is an array, so for the calculation of pagerank, we flattened the references array then did group by on references column with aggregation of ($PR(p_j) / L(p_j)$) corresponding to each paper id in references column. This gave us new dataframe with paper id and updated pagerank value.
3. Finally merged this newly created dataframe with old dataframe to update pagerank values.
4. Repeated above 3 steps five times, after that pagerank value got saturated.

5) Author Count:
The field ‘author’ contained a list of authors who wrote that particular paper. We extracted the number of authors in that list and added it as a separate feature to monitor if the number of authors has any effect on the quality of paper.

6) References count:
   Similar to authors, our dataset contained the list of references (citations) made by that paper. For our baseline model, we extracted the number of references made as a feature to check if more references made contributes to a better quality.

7) Paper Importance:
   a) We created a new feature from citation count: Paper Importance
   b) Calculated as sum of citation count of all papers, which cited that paper.
   c) This is a good feature to predict rank, as more the citation count of a paper, more credit should be given to the paper being cited by this paper.

8) H-index [1]:
   In the dataset, we have author array corresponding to each paper. For the calculation of h-index, we first flattened the author array to get new dataframe. Then we grouped this new dataframe on authors, to get citation details of each author. Applying user-defined function on citation count details of grouped dataframe, we got h-index of authors. User-defined function, takes citation count list as input, then sort this list in decreasing order, then iterate over this sorted list to get last position in which citation count is greater than or equal to the position (this position is h-index).

9) Age of the paper:
   Number of years since it was published. This is important factor to consider in calculating rank of a paper. Because if a paper was published long ago, it had more time to receive citation, while papers which got published recently will have less time to receive citation. So, paper could be less popular because it came recently, but that does not necessarily mean that paper should be given less rank.

10) Average citation count at venue :
   Rank of paper also depends on the conference it is published, as some conferences have strict criteria of paper selection. That factor is considered by calculating average citation count of the paper published at that conference.

   'venue_citation_avg' = 'venue_citation_sum' / 'venue_citation_count'

   a. 'venue_citation_sum' = Sum of citation count of all papers published at a venue
   b. 'venue_citation_count' = Total papers published at a venue.
   c. 'venue_citation_avg' = Average citation count of a paper published at a venue

11) Average citation count in domain :
   Rank of paper depends on the domain of the paper. There can be some domain in which there is more scope of paper getting published, while in some its very difficult. That factor is considered by calculating average citation count of the paper published in a domain.

   'domain_citation_avg' = 'domain_citation_sum' / 'domain_citation_count'

Ranking Metric (based on the above features):

Paper Ranking Score for a paper ***'p'*** is defined as :

$$rank(p) = w_1 * PR(p) + w_2 * c(p) + w_3 * (T(p) + V(p) + CS(p) + R(p) + I(p)) - w_4 * (a(p) + D(p))$$

w_1=10, w_2=5, w_3=2, w_4=2	
PR(p)	Page Rank value of paper 'p'
c(p)	Total Citation Count of paper 'p'
a(p)	Age of paper 'p'
T(p)	Time factor score of paper 'p'
V(p)	Average citation count of venue in which this paper is published
D(p)	Average citation count of domain of paper 'p'
CS(p)	Citation sentiment score of paper 'p'
R(p)	Reach value of paper 'p'
I(p)	Importance of paper 'p'

Figure X : Definition of parameters used in the metric

Rationale behind the construction of the metric :

1. We observed that some metrics like pagerank and citation count were very large and some parameters like Time factor score were very small. Hence before plugging into the ranking metric, we first normalized the scores.
2. We observed that the quality of paper greatly depends on Pagerank and Citation Count and hence we gave a high weight to them in our metric.
3. Citation sentiment, Venue, Reach value and Importance also positively influences our ranking score albeit not that much, hence W_3 is small.
4. We observed that more the age of the paper or the number of papers in that domain, naturally the citations would be more. To be fair to those papers which are recently published and/or are published in domains that don't have a lot of papers, we gave a negative weight to these parameters.
5. We finally divide by 4 to average out the impact of the parameters.
6. Finally we have determined the values for the weights W_1, W_2, W_3 and W_4 empirically.

Author Ranking Score for an author ‘a’ is defined as :

$$rank(a) = (1/n) * \sum_{p=1}^{n} rank(p)$$

a	author
p	paper
n	Total number of papers published by author

I.e. we define it as the average of the paper rank score of all the papers that this author has published.

For example, if an author A has published two papers P_1 and P_2 such that P_1 has score of 100 and P_2 has score of 50 then the rank score of author A will be (100+50)/2 = 75.

Tasks:

1. Rank top 100 and compare against h-index:

Author	h_index	Ranking_score	Discipline	Rank
Christos Faloutsos	99	1161.520397	Data science	1.0
Berthold K. P. Horn	7	1117.636170	Data science	2.0
Alexander J. Smola	84	1111.652414	Data science	3.0
Brian G. Schunck	4	816.151113	Data science	4.0
Hector Garcia-Molina	133	756.780836	Data science	5.0
Thomas E. Anderson	25	652.914607	Data science	6.0
Stefan Savage	78	613.385287	Data science	7.0
Michalis Faloutsos	61	566.949415	Data science	8.0
Carl Kesselman	79	565.213136	Data science	9.0
Ion Stoica	119	541.258469	Data science	10.0
Nigel Davies	54	469.150154	Data science	11.0
Thomas L. Magnanti	34	462.912587	Data science	12.0
Wolfram Burgard	101	449.306671	Data science	13.0
Randy H. Katz	92	418.388894	Data science	14.0
John N. Tsitsiklis	71	355.811357	Data science	15.0
Holger H. Hoos	63	310.815823	Data science	16.0
Rajkumar Buyya	121	304.795538	Data science	17.0
Philip S. Yu	149	289.061411	Data science	18.0
Mario Gerla	118	245.506443	Data science	19.0
Kai Chen	20	151.564618	Data science	20.0

Figure 7: TOP researchers in Data Science Domain

In this dataframe, we show the top 20 researchers in the Data Science domain according to our ranking metric. As we can see, Most of the authors which have high h-index also have high rank according to our metric. However there are several exceptions like Berthold K.P. Horn and Brian G. Schunck, who have high rank according to our metric but lower h-index. This can be attributed to several factors which h-index does not take care of.

Author	h_index	Ranking_score	Discipline	Rank
Thomas G. Dietterich	71	1196.113987	Computational Geometry	1.0
Daphne Koller	131	1180.739357	Computational Geometry	2.0
Philip A. Bernstein	76	881.935901	Computational Geometry	3.0
John R. Koza	9	853.141317	Computational Geometry	4.0
Edmund M. Clarke	101	836.400604	Computational Geometry	5.0
Terry Winograd	60	752.561708	Computational Geometry	6.0
David Gefen	55	712.583676	Computational Geometry	7.0
Moni Naor	87	674.594621	Computational Geometry	8.0
Serge Abiteboul	81	617.460778	Computational Geometry	9.0
Moshe Y. Vardi	100	491.765165	Computational Geometry	10.0
Salvatore J. Stolfo	90	414.460714	Computational Geometry	11.0
Pedro M. Domingos	84	398.663853	Computational Geometry	12.0
Joachim Weickert	70	379.734267	Computational Geometry	13.0
Michael Wooldridge	89	318.700139	Computational Geometry	14.0
Yoshua Bengio	131	313.588467	Computational Geometry	15.0
Thomas A. Henzinger	102	217.041434	Computational Geometry	16.0
Kim Guldstrand Larsen	77	198.879500	Computational Geometry	17.0
Ronald R. Yager	116	195.220920	Computational Geometry	18.0
Didier Dubois	124	131.990314	Computational Geometry	19.0
Henri Prade	121	126.766557	Computational Geometry	20.0

Figure 8: TOP researchers in Computational Geometry Domain

In this dataframe, we show the top 20 researchers in the Computational Geometry domain according to our ranking metric. Here too, most of our high ranked authors correlate with their h-index. Exceptions include John R. Koza and David Gefen who have high ranks according to us but their h-index is low. Reason is same: that is, h-index doesn't take care of factors like time analysis, citation sentiment analysis etc which our ranking metric does.

Author	h_index	Ranking_score	Discipline	Rank
Ronald L. Rivest	90	2625.492943	Algorithms	1.0
Fred D. Davis	41	1317.525037	Algorithms	2.0
Paul C. van Oorschot	71	1169.962642	Algorithms	3.0
Ron Kohavi	54	899.749210	Algorithms	4.0
Jennifer Widom	97	837.005562	Algorithms	5.0
Ian Horrocks	91	828.965326	Algorithms	6.0
Prabhakar Raghavan	13	775.733972	Algorithms	7.0
Sebastian Thrun	142	720.847856	Algorithms	8.0
Rakesh Agrawal	103	688.615248	Algorithms	9.0
Advaith Siddharthan	24	566.054971	Algorithms	10.0
Rudolf Wille	22	554.038533	Algorithms	11.0
Martin Ester	57	543.614971	Algorithms	12.0
John Riedl	17	542.369031	Algorithms	13.0
M. N. Murty	20	530.699786	Algorithms	14.0
Jiirg Sander	2	471.718980	Algorithms	15.0
Xu Xiaowei	30	461.332024	Algorithms	16.0
H.-P. Kriegel	91	460.178149	Algorithms	17.0
Hinrich Sch√°tze	11	421.482298	Algorithms	18.0
Nicholas R. Jennings	50	316.390081	Algorithms	19.0
Wei Wang	126	-179.360123	Algorithms	20.0

FIGURE 9: TOP researchers in Algorithms Domain

In this dataframe, we show the top 20 researchers in the Algorithms domain according to our ranking metric. As we can see,There are some authors like Jirg Sander and Prabhakar Raghavan whose h-index does not correspond with our ranking metric. Reasons are same as above.

Author	h_index	Ranking_score	Discipline	Rank
John E. Hopcroft	57	2752.775107	Computational Biology	1.0
Robert E. Schapire	75	1481.210967	Computational Biology	2.0
Luc J. Van Gool	125	1390.815413	Computational Biology	3.0
Andrew Zisserman	145	1351.951449	Computational Biology	4.0
Bernhard Sch√∂lkopf	126	1232.742743	Computational Biology	5.0
Tinne Tuytelaars	54	1009.491141	Computational Biology	6.0
Gerard Salton	66	1003.872531	Computational Biology	7.0
Christopher D. Manning	108	717.860161	Computational Biology	8.0
Biing-Hwang Juang	48	578.859808	Computational Biology	9.0
Lawrence R. Rabiner	81	574.617721	Computational Biology	10.0
Patrick J. Flynn	60	533.127933	Computational Biology	11.0
A.K. Jain	179	531.939641	Computational Biology	12.0
Nicholas Ayache	101	508.948296	Computational Biology	13.0
Andrew Y. Ng	112	469.556690	Computational Biology	14.0
Antonio Torralba	87	452.097873	Computational Biology	15.0
Anil K. Jain	179	389.373512	Computational Biology	16.0
Mubarak Shah	100	386.273420	Computational Biology	17.0
Thomas S. Huang	144	381.231218	Computational Biology	18.0
Jian Sun	76	330.797419	Computational Biology	19.0
Lei Zhang	95	251.224995	Computational Biology	20.0

FIGURE 10: TOP researchers in Computational Biology Domain

In this dataframe, we show the top 20 researchers in the Algorithms domain according to our ranking metric. Surprisingly there are no or few exceptions here. However, the metric does not exactly correlate with the h-index. As we can see highest ranked author is John E. Hopcroft but highest h-index is of Anil K. Jain. This is because our metric considers many different factors than h-index does and is better.

2. Network Reach:

It is found that a paper often gets citations from people who were co-authors or are close to the original author (eg: friend of friend). This can create a bias as old researchers may have formed a very good social network and any work they publish can gain high visibility due to word of mouth advertising. In contrast, it may be very well be the case that a young researcher may have published extremely high quality work but he doesn't know anybody in the research community yet. Due to this, his work will be read by other authors only when they come to access it via some search engine for scholarly literature such as Google Scholar. To avoid this unfairness, in our metric, we have ranked a paper more highly if it was cited by an author who is far from the original author in the author collaboration graph.

2.1 *Author Collaboration Graph***:** [11,12]

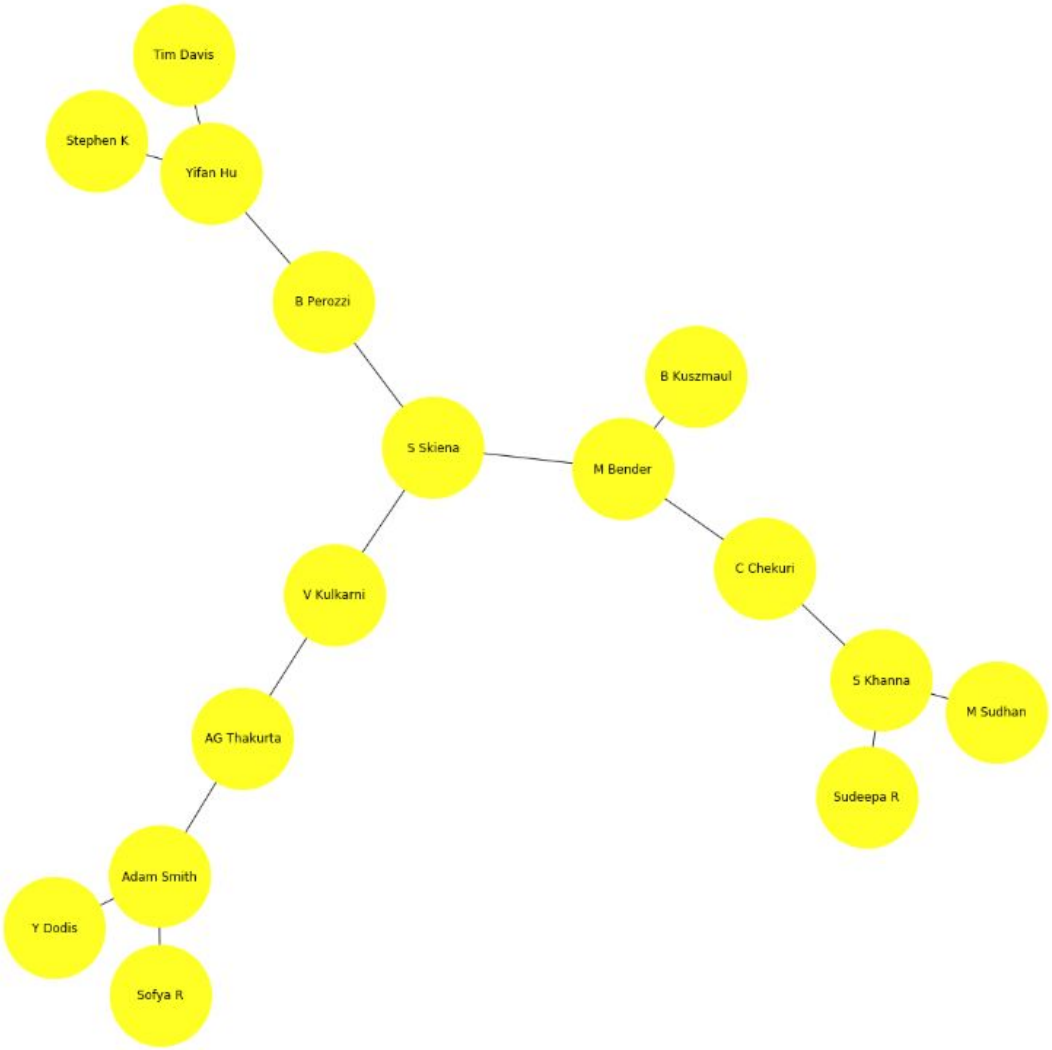


Figure 11 : Author Collaboration Graph of a subset of authors in our dataset

2.2 *Graph Construction:*
We have constructed the author collaboration graph (aka co-authorship network) as follows:

1. Each node represents a unique author
2. Each edge represents co-authorship between the two authors for any publication
3. Each edge is unit weight

2.3 *Collaboration Distance : [12]*
We have defined it as the shortest path distance between two authors
For example,

a. the collaboration distance between ‘S Skiena’ and ‘M Bender’ is 1. They are directly connected.
b. the collaboration distance between ‘S Skiena’ and ‘Tim Davis’ is 4. It is the path {‘S Skiena’ - ‘B Perozzi’ - ‘Yifan Hu’ - ‘Tim Davis’ }
c. the collaboration distance between ‘Sofia R’ and ‘M Sudhan’ is 8

2.4 *Paper Reach:*
We have defined it as the maximum collaboration distance between any author of given paper and any author of any of its citations.

1) Create a set 'A' that contains all authors of given paper
2) Create a set 'B' that contains find all authors from all the citations of the given paper
3) find the collaboration distance between each author of set A and B.
4) return the maximum one

Sample Calculation of reach of a paper based on above graph:
Assume Paper A has the Authors : ['S Skiena', 'V Kulkarni'] and has two citations B and C.
Assume Citation B has the Authors : ['Sudeepa R']
Assume Citation C has the Authors : ['Adam Smith','Sofiya R']

Collaboration Distance between 'S Skiena' and 'Sudeepa R' is : 5
Collaboration Distance between 'S Skiena' and 'Adam Smith' is : 3
Collaboration Distance between 'S Skiena' and 'Sofiya R' is : 4

Collaboration Distance between 'V Kulkarni' and 'Sudeepa R' is : 6
Collaboration Distance between 'V Kulkarni' and 'Adam Smith' is : 2
Collaboration Distance between 'V Kulkarni' and 'Sofiya R' is : 3

Reach of paper A : MAX{ 5,3,4,6,2,3 } = 6.

***2.5** Domain Reach of a Paper :*

We can define it as the total number of unique domains of citations of the paper.

1. First we have extracted domain of each paper using topic modelling.
2. Second we have calculate domain of each author as the union of domains of all his papers.
3. Now while computing the paper reach, we keep a count of unique domains of authors in set B.

Note : In our ranking metric, we have used the simple paper reach instead of domain reach.

3. <u>Compare researchers with substantial between our metric and h-index:</u>

We identified top 20 authors (independent of domain) as per our ranking metric and compared it against their h-Index.

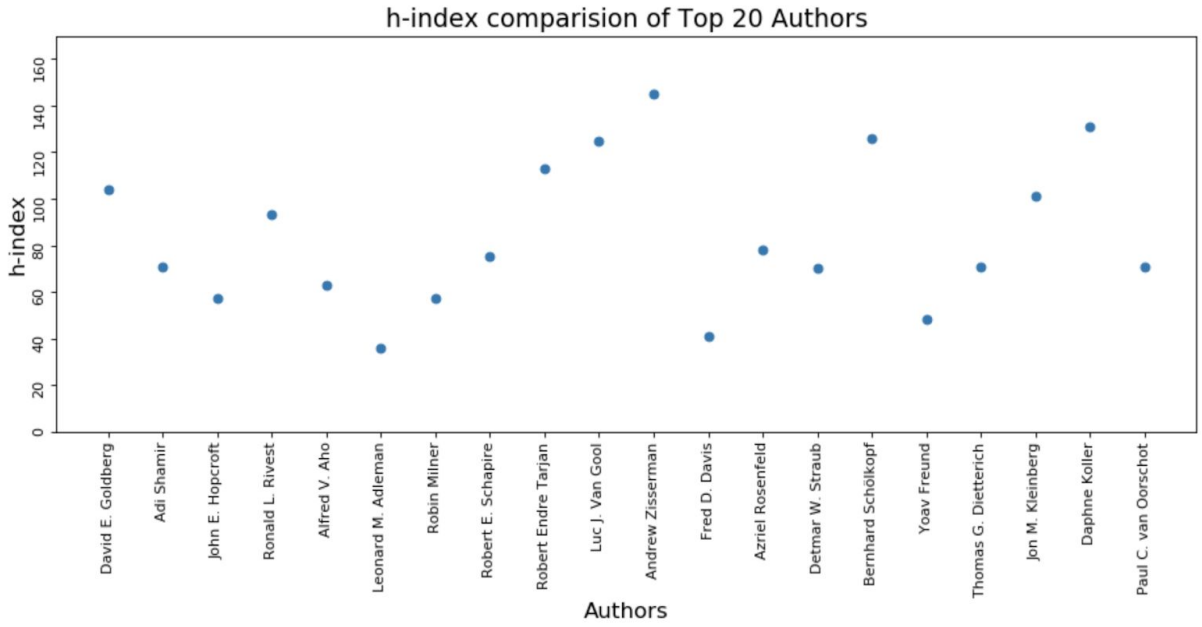


Figure 12: Top 20 Authors (leftmost is 1st rank, rightmost is 20th rank) and their respective h-Index

***3.1** Topmost author as per our ranking metric:*

'**David E Goldberg**' is ranked the topmost author as per our ranking metric. We have shown below the top 5 papers of this author as per our ranking metric.

authors	id	ranking_score	pageRankValue	n_citation	citation_sentiment	domain_citation_avg	Topic	venue_citation_avg	year
David E. Goldberg	6a6b9aa6-683f-4c7c-b06e-9c3018d10fd3	3661.244512	54643.636990	73362	1613	54.804301	Mobile computing	54.262526	1989
David E. Goldberg	fbbddfd2-5b7c-4f54-8718-5fe1f58ebf33	187.950815	4842.159256	1641	41	54.804301	Mobile computing	54.262526	1989
David E. Goldberg	32db0b6b-7326-4bd6-9404-fa88ce9e0746	161.969638	2637.364396	2923	70	54.804301	Mobile computing	91.567568	1994
David E. Goldberg	b091bbe4-2329-482b-a949-d27fac83f381	126.967706	1815.795924	2614	49	53.176394	Linear Algebra	161.774648	1991
David E. Goldberg	2b09da78-a6a1-437d-a7b3-c71d634b13d6	34.534246	153.534293	1572	13	54.583123	Computer Networks	45.551724	1998

Figure 13: Top 5 papers of 'David E Goldberg'

3.2** Analysis:**

1. He has published many good quality papers across multiple domains
2. Most of his papers have a very high page rank score that means they are cited by papers which are themselves of good quality.
3. In general, his papers have a very high citation count compared to the average citation count of the domain of that paper
4. In general, his papers have a very high citation count compared to the average citation count of the venue in which that paper was published
5. In general, his papers have a very high citation sentiment score.

His h-index is 104 which is quite high and thus aligns with our result.

***3.3** Authors with substantial difference between our ranking metric and h-Index:*

We have identified '**Leonard M. Adleman**' as one such author. He is **ranked as 6th** as per our metric but his **h-Index is 36**. Below are his top 5 papers as per our ranking metric :

authors	id	ranking_score	pageRankValue	n_citation	citation_sentiment	domain_citation_avg	Topic	venue_citation_avg	year
Leonard M. Adleman	3fb43b00-905c-4a08-934d-198ea4eb66c3	1895.095585	47955.353140	18861	486	53.176394	Linear Algebra	171.915351	1978
Leonard M. Adleman	239eccc0-3d4c-4e62-a1ba-2e8722bc5640	311.816471	265.543663	226	11	54.804301	Mobile computing	118.521851	2001
Leonard M. Adleman	79e4a359-8d9e-40bd-a986-5246279acc65	143.478542	181.355500	163	6	54.804301	Mobile computing	135.916803	1993
Leonard M. Adleman	beb8343c-8e4f-40d3-bf94-04bb7abbae4c	113.342800	184.692623	138	3	54.804301	Mobile computing	54.262526	2000
Leonard M. Adleman	01cf1525-68a3-47af-b976-0ed66dc13acb	100.918135	130.289360	139	2	54.804301	Mobile computing	118.521851	1976

Figure 14 : Top 5 papers of 'Leonard M. Adleman'

We believe the result of **our metric is better than h-Index** due to below **reasons**:

1. We can see that he has published good quality papers in multiple domains
2. In general, his papers have a very high citation count compared to the average citation count of the domain of that paper
3. In general, his papers have a very high citation count compared to the average citation count of the venue in which that paper was published

Another such author that we have identified is **'Fred D. Davis'**. He is **ranked as 12th** as per our metric but his **h-Index is 41**. Below are his top 5 papers as per our ranking metric:

authors	id	ranking_score	pageRankValue	n_citation	citation_sentiment	domain_citation_avg	Topic	venue_citation_avg	year
Fred D. Davis	9d912297-e52f-4ab6-add4-633e0f263933	1307.541050	14393.573360	27068	830	58.304374	Algorithms	382.608961	1989
Fred D. Davis	9fdf7736-2363-4c03-8833-31e2dcb17fbf	118.965955	831.646987	378	11	55.262050	Image Processing	177.200418	2003
Fred D. Davis	190a1168-0d2c-43db-8462-5fe58c001fbb	103.581204	531.790843	264	5	55.809836	Computational Biology	177.200418	2014
Fred D. Davis	8b123cb5-8fb3-4e61-8a77-9fafcbd10df7	92.369317	409.239158	248	4	58.304374	Algorithms	22.814126	2011
Fred D. Davis	fee5526e-8c32-40c8-abb4-74a8fb926c6f	90.193855	389.091862	124	3	54.583123	Computer Networks	77.777778	2015

Figure 15 : Top 5 papers of 'Fred D. Davis'

We believe the result of **our metric is better h-Index** due to below reasons:

1. We can see that he has published good quality papers in multiple domains
2. In general, his papers have a high page rank that means they were cited by papers that are themselves of good quality
3. In general, his papers have a very high citation count compared to the average citation count of the domain of that paper
4. In general, his papers have a very high citation count compared to the average citation count of the venue in which that paper was published

4. Time analysis of paper citation count against our metric:

Time factor score = ('n_citation')/('year_citation_sum' * 'year_citation_max')

Where,
year_citation_count = Calculated total papers published in a year
year_citation_sum = Sum of citation count of all papers published in a year
'n_citation' = Citation count of a paper

For performing time analysis of the papers, we extracted features like domain of the paper (using LDA topic modelling) and Time factor score (calculated using the citation count and year given in the original dataset).
We have also taken the age of the paper into account. This we extracted using the year column given in the original dataset (by subtracting that year with the current year). The intuition behind this is that the papers that were published earlier will naturally will have more number of citations and hence our metric should smooth that out.
This is how we use the above factors to perform time analysis and smooth out factors like technology boom, etc.
Consider an example:
After 2012, in NLP domain, neural networks emerged as a powerful machine learning tool which lead to many publications in this domain. This does not necessarily mean all the papers published during this time, which received high citations, were necessarily high quality work.

ANALYSIS:
I'll explain my analysis with the help of an example:
Consider the following two papers from NLP domain:
PAPER 1: Consider the following dataframe:

	authors	id	n_citation	title	year	year_citation_sum	year_citation_max	age	Topic	time_factor
35144	[Rudolf Krawczyk]	8fce594c-da33-4ac4-8761-948425495542	50	Verbesserung von Schranken für Eigenwerte und ...	1970	7049	982	48	NLP	72.232227

Figure 16 : Most popular paper of NLP domain published in 1970

This is a paper from NLP domain that was published in 1970. Infact, this is the most popular paper of NLP domain from the year 1970, that is the paper with maximum number of citations in 1970 in NLP. NLP domain was not very popular at that time and hence we can see from the above data frame, that the year_citation_sum and year_citation_max is relatively low. However, the time_factor score (after scaling, multiplying by 10^7) that we calculate for the above paper is very high (=72.23) although the n_citation of the paper is just 50 (relatively low)

PAPER 2: Now consider the following paper

	abstract	authors	id	n_citation	title	year	year_citation_sum	year_citation_max	age	Topic	time_factor
39468	A common problem with designing and developing...	[Marianna Obrist, Sue Ann Seah, Sriram Subrama...	02971f2f-01b0-4330-ab05-78b330fb656b	2233	Talking about tactile experiences	2013	845321	6186	5	NLP	4.270288

Figure 17 : Most popular paper of NLP domain published in 2013

This is the most popular paper from 2013 from NLP domain (paper with maximum number of citations in 2013 in NLP). Remember that this was the time when the technology boom of Deep learning (Neural networks) started in NLP. Hence a lot of papers were published and cited from this time. As we can see from the above dataframe, year_citation_sum and year_citation_max are very high for 2013 for NLP domain. However, the time factor score (after scaling, multiplying by 10^7) that we give to this paper is low (=4.27) although the individual citation count of the paper is very high (=2233).

In this way, with the help of domain of the paper and time factor score, we are ensuring that our metric incorporates time factors like technology boom that to smooth out papers which got higher citation counts because of factors other than solely the quality of material presented.

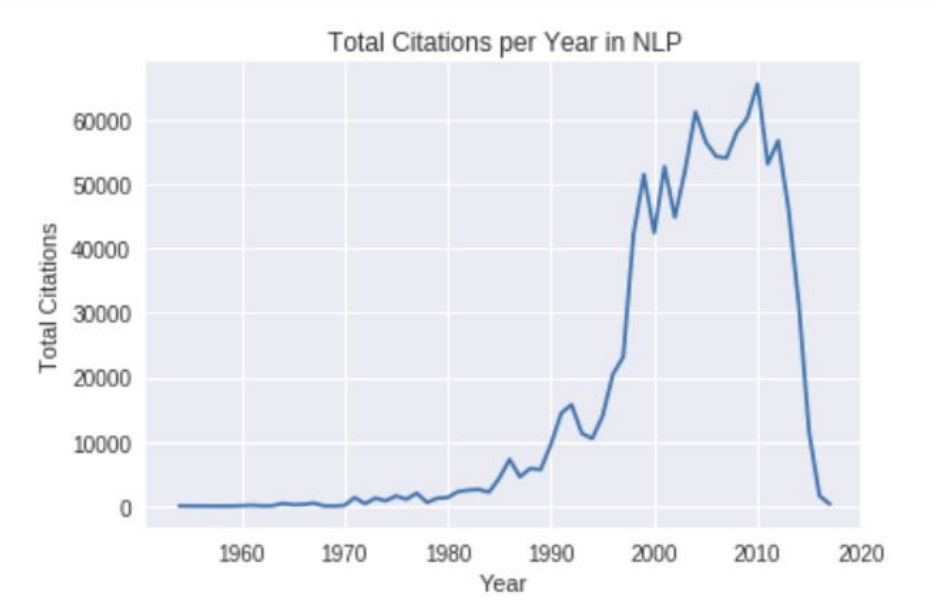


Figure 18 : Total Citations per year in NLP domain

We can clearly see from the above graph, the technology boom in NLP.

5. Predicting popular papers on arxiv:

For the predictions of papers on Arxiv, we used kaggle dataset[14]

In the dataset we found following useful features:

1. Author(s) of the paper
2. Summary of the paper
3. Title of the paper
4. Year, which can be used to analyse time factor

1. Author(s) of the paper: We have authors of the paper in the arxiv dataset. We have the author score of all the authors in our training set, created by our original ranking metric, of the authors. Using that we got the author score for the authors of this arxiv dataset. If author is not there in training data, we take score as 0.0.
2. Summary of the paper: From the summary of the paper, we got the sentiment score of the paper, which will be another feature for rank prediction
3. Title of the paper: From the title of the paper, we got the domain of the paper. For each of our identified domain we have the 'domain_avg_citation' value, which considers the impact of domain on the quality of paper. We used that as a feature for prediction of score of the paper.
4. Year: We get the age of the paper from its year.

In addition to above features we extracted many other features (described in the training data below) and then determined the score of the paper, which tells whether paper should become popular or important.

5.1 TRAINING OUR MODEL

	id	authors	author_count	sentiment	Topic	domain_citation_avg	year	ranking_score	author_score	age
0	001c8744-73c4-4b04-9364-22d31a10dbf1	[Altaf Hossain, Faisal Zaman, Mohammed Nasser,...	4	1	1	57.087180	2009	-1.309793	-3.216079	9
1	00bcf2d5-1592-46b0-81fd-933f90b5ecca	[Alvaro L. Islas, Constance M. Schober]	2	0	8	54.804301	2002	-1.254513	-2.190841	16
2	00c85316-bddf-4bcb-93f5-097adadd73c2	[Patrick Cousot, Radhia Cousot]	2	1	2	55.424089	1991	-2.215803	38.787464	27
3	010d4ce9-0279-4166-ae73-14551ded6404	[John R. Douceur, Jeremy Elson, Jon Howell, Ja...	4	1	9	52.136057	2008	16.544018	130.063854	10
4	012b88ae-a763-45d6-8f19-2ec9ff739d5f	[Carmen Fernández-Daza Álvarez, Philippe Langl...	3	-1	5	55.809836	2004	-1.691160	-13.671207	14

Figure 19: Training data

TRAINING PROCESS:

1. First we create Training data for our model using the above mentioned features. This we did by extracting the relevant features(which can be found in the test set) from the main dataframe. This training data frame included author_count, sentiment, Topic, domain_citation_avg, author_score, age . (X labels)
2. We kept our ranking score as the prediction metric (Y label).
3. We then splitted the data into training and testing data (80% and 20%).

LINEAR REGRESSION

4. We then applied linear regression as our baseline model to train the data. The results however were not very promising which is expected since this is a very simple model to predict a very complicated problem.

LGBM

5. Finally we apply LGBM to train our model with the following parameters:

```
def run_lgb(train_X, train_y, val_X, val_y):
    params = {
        "objective" : "regression",
        "metric" : "rmse",
        "num_leaves" : 30,
        "min_child_samples" : 100,
        "learning_rate" : 0.1,
        "bagging_fraction" : 0.7,
        "feature_fraction" : 0.5,
        "bagging_frequency" : 5,
        "bagging_seed" : 2018,
        "verbosity" : -1
    }
```

The results this time were a lot better with an RMSE score of around `1.6079`.This shows that our trained model is very good and is able to predict the ranking score (and hence the rank) of the paper very accurately.

Before applying our model to the arxiv dataset, we first plot the feature importance graph in LGBM and visualise the features that our model gives most importance to, while calculating the ranking score:

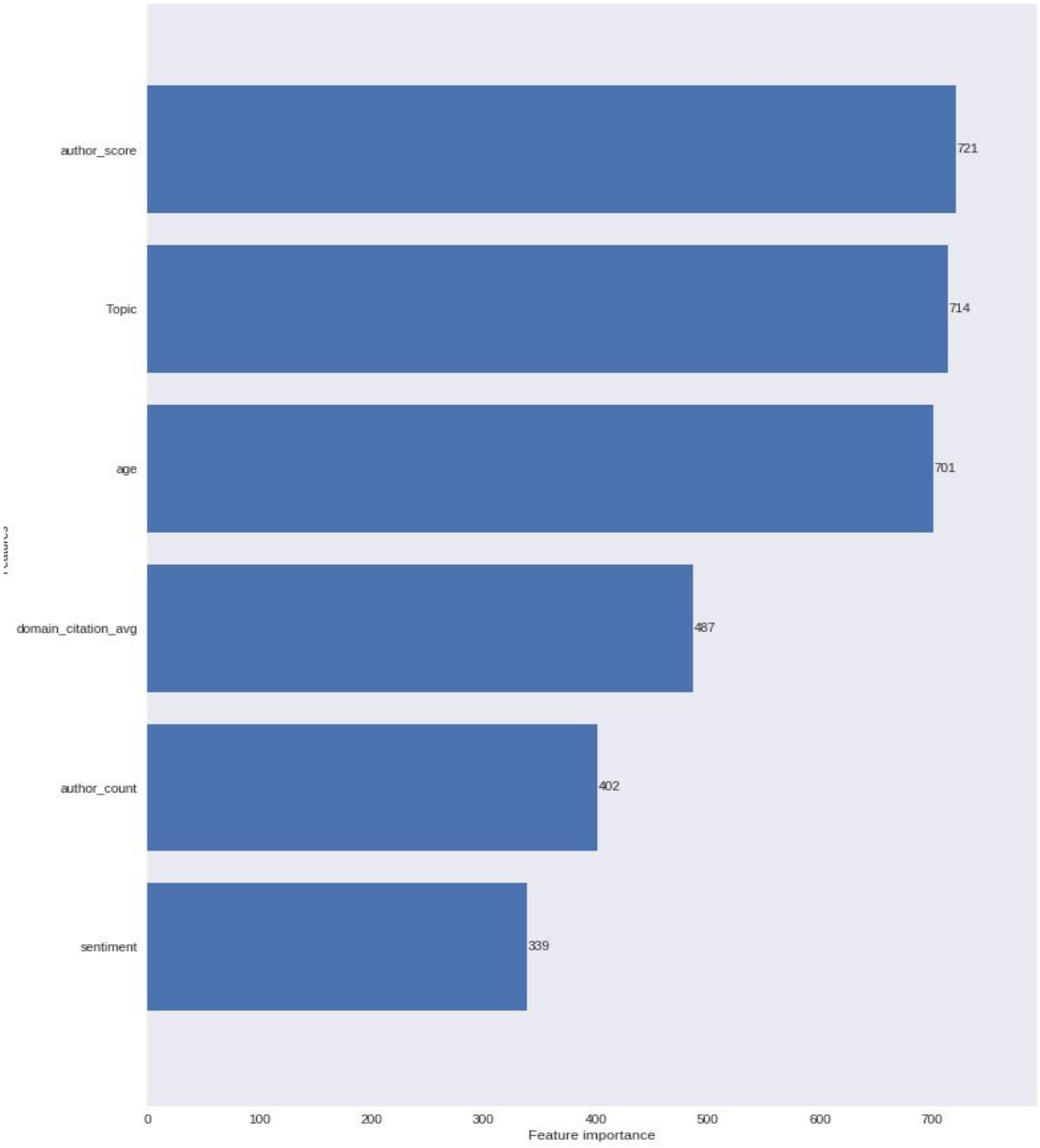


Figure 20: LGBM feature importance

Inferences from the above plot:

1. The author score gets the highest importance which is great since that is one thing which we can easily calculate in the papers from arxiv and one that is definite. This shows that the popularity of the paper would depend on who is writing it.

2. Next the popularity of the paper would depend on the domain in which it is published. Some domains (eg. NLP) have a lot of research scope and hence its papers would naturally have more citations in future.
3. In addition to the above two, the popularity of the paper would also minorly depend on its age (that is the year in which it is published), domain citation average, author_count (number of authors collaborating) and sentiment (extracted using the summary in kaggle dataset).

TESTING OUR MODEL (on arxiv dataset obtained from kaggle[14]):

author	day	id	link	month	summary	tag	title	year
[{'name': 'Ahmed Osman'}, {'name': 'Wojciech S...	1	1802.00209v1	[{'rel': 'alternate', 'href': 'http://arxiv.or...	2	We propose an architecture for VQA which utili...	[{'term': 'cs.AI', 'scheme': 'http://arxiv.org...	Dual Recurrent Attention Units for Visual Ques...	2018
[{'name': 'Ji Young Lee'}, {'name': 'Franck De...	12	1603.03827v1	[{'rel': 'alternate', 'href': 'http://arxiv.or...	3	Recent approaches based on artificial neural n...	[{'term': 'cs.CL', 'scheme': 'http://arxiv.org...	Sequential Short-Text Classification with Recu...	2016
[{'name': 'Iulian Vlad Serban'}, {'name': 'Tim...	2	1606.00776v2	[{'rel': 'alternate', 'href': 'http://arxiv.or...	6	We introduce the multiresolution recurrent neu...	[{'term': 'cs.CL', 'scheme': 'http://arxiv.org...	Multiresolution Recurrent Neural Networks: An ...	2016
[{'name': 'Sebastian Ruder'}, {'name': 'Joachi...	23	1705.08142v2	[{'rel': 'alternate', 'href': 'http://arxiv.or...	5	Multi-task learning is motivated by the observ...	[{'term': 'stat.ML', 'scheme': 'http://arxiv.o...	Learning what to share between loosely related...	2017
[{'name': 'Iulian V. Serban'}, {'name': 'Chinn...	7	1709.02349v2	[{'rel': 'alternate', 'href': 'http://arxiv.or...	9	We present MILABOT: a deep reinforcement learn...	[{'term': 'cs.CL', 'scheme': 'http://arxiv.org...	A Deep Reinforcement Learning Chatbot	2017

Figure 21: Testing data (from the kaggle arxiv dataset)

TESTING PROCESS:

1. We calculate author_count, sentiment, Topic, domain_citation_avg, author_score, age using the features using the same process as in training data.

2. Then we apply our above trained model to it to predict the ranking score of the papers in arxiv.

3. The more the score, the higher its chances of becoming popular in future.

PREDICTING TOP 10 PAPERS FROM ARXIV THAT WILL BECOME POPULAR

After we have predicted the ranking score of the papers using our above trained model, then we take the top 10 papers, which have the highest ranking score which have the most chance of becoming popular. These are as below:

id	summary	title	year	ranking_score	Rank
1801.06700v1	We present MILABOT: a deep reinforcement learn...	A Deep Reinforcement Learning Chatbot (Short V...	2018	57382.6728	1.0
1705.08142v2	Multi-task learning is motivated by the observ...	Learning what to share between loosely related...	2017	54329.6492	2.0
1709.02349v2	We present MILABOT: a deep reinforcement learn...	A Deep Reinforcement Learning Chatbot	2017	47839.2829	3.0
1709.08878v1	We propose a new generative model of sentences...	Generating Sentences by Editing Prototypes	2017	46292.3893	4.0
1802.00209v1	We propose an architecture for VQA which utili...	Dual Recurrent Attention Units for Visual Ques...	2018	43145.3244	5.0
1609.06492v1	The paper introduces a new method for discrimi...	Document Image Coding and Clustering for Scrip...	2016	41030.7383	6.0
1610.01076v1	Together with the development of more accurate...	Tutorial on Answering Questions about Images w...	2016	39028.4630	7.0
1705.07962v2	Transforming a graphical user interface screen...	pix2code: Generating Code from a Graphical Use...	2017	38293.4739	8.0
1606.00776v2	We introduce the multiresolution recurrent neu...	Multiresolution Recurrent Neural Networks: An ...	2016	37823.6731	9.0
1603.03827v1	Recent approaches based on artificial neural n...	Sequential Short-Text Classification with Recu...	2016	24066.8732	10.0

Figure 22: Top 10 arxiv papers with most popularity chance

Inferences:

1. As we can see, the top most paper (and many others) of Deep learning which is understandable, since its one of the ‘hot topics’ these days in NLP.

2. Reinforcement learning in Data science is another field which is a very popular domain nowadays.

3. Image coding (Ranked 6) is also predicted to be popular, which is due the authors that have written it, are themselves very popular.

References:

[1]https://en.wikipedia.org/wiki/H-index
[2]https://www.researchgate.net/publication/220364697_The_discipline_dependence_of_citation_statistics
[3]https://www.sciencedirect.com/science/article/pii/S1751157710000970#sec0010
[4]https://lfs.aminer.cn/lab-datasets/citation/dblp.v10.zip
[5]https://pypi.org/project/scholarly/
[6]http://eigenfactor.org/projects/journalRank/rankings.php?bsearch=<JournalName>&searchby=journal&orderby=eigenfactor
[7]https://libraries.mit.edu/scholarly/publishing/apis-for-scholarly-resources/
[8]https://towardsdatascience.com/topic-modeling-and-latent-dirichlet-allocation-in-python-9bf156893c24
[9]https://aws.amazon.com/getting-started/tutorials/analyze-sentiment-comprehend/
[10]http://ilpubs.stanford.edu:8090/422/
[11]https://en.wikipedia.org/wiki/Collaboration_graph
[12]https://www.researchgate.net/publication/273312860_Co-authorship_networks_A_review_of_the_literature
[13]https://en.wikipedia.org/wiki/Collaboration_graph#Collaboration_distance
[14]https://www.kaggle.com/neelshah18/arxivdataset