ENS 491/2 - Graduation Project
Final Report

Understanding the Academic World

Submitted by
Berkan Teber

Under the supervision of
Kamer Kaya
Hüsnü Yenigün

2017 - 2018

Sabancı University
Faculty of Engineering and Natural Sciences

1 Executive Summary

In this project, our objective was having a better understanding of the academic world. To achieve this goal we have evaluated different components of the academic world with respect to both current methods and modified versions of these methods. We have evaluated 3 components: (i) papers which are essentially the product of the academic world, (ii) authors of these papers who can be considered as the main input of the academic world, and (iii) fields with some common behavior differentiating themselves from the others.

To evaluate papers and authors, we have used both existing methods that are being used vastly such as h-index and some Random Walk based methods which are relatively new to measure the academic world. To evaluate fields, we have investigated some of the statistics for each field.

In our work, we have shown that there are some important differences both within existing indexes and between the existing ones and our own evaluations. We have also identified some characteristics of different fields of the academic world.

2 Problem Statement

In this project, we aimed to have a better understanding of the academic world. To achieve this goal, we have identified several questions that we think answering them would help us understand the academic world much better. These questions are (i) "Which papers are the most influential in their field?", (ii) "Which researchers are more successful?", and (iii) "Which scientific areas are the most promising?".

In order to answer these questions, we have divided our problem into 3 subproblems: evaluating papers, evaluating authors and evaluating fields. Furthermore, we have been divided into 2 groups and focused on different aspects. We have focused on the existing evaluation techniques and their comparison, and comparison of different fields, while the other group has focused on developing a new evaluation technique based on Random Walk.

2.1 Objectives and Tasks

2.1.1 Constructing Graphs

In this project, we have used a huge paper-citation graph from 2 different datasets to represent the academic world. AMiner dataset [1] had 154,771,162 papers while Microsoft Academic Graph [2] had 166,192,182 papers.

As the first step, we have filtered the information on these datasets, so that we can have a relatively small dataset with necessary informations only. For this purpose, for each paper, we have filtered the following information only and represented them in a more systematical way:

1. Source file
2. Year
3. Authors with their affiliations
4. References

As a result of this step, we have produced a graph for each dataset with the above information only.

As the second step we have merged these 2 previosly constructed graphs into one. There were 64,398,608 one-to-one matchings between 2 original datasets and we have used these matching information to merge the graphs. While merging the graphs, if a paper is available in only one dataset, we have used only that information. On the other hand, if a paper is available on both datasets, we have taken the union of its references in each dataset.

As a result, we have obtained a single merged graph with necessary information only.

2.1.2 Constructing CRS-Formatted Graphs

In this step, we have constructed several CRS formatted graphs.

CRS (Compressed Row Storage) is a widely used format to represent sparse graphs. Given a graph $G = (V, E)$, CRS consists of 3 arrays as follows:

1. Row Pointer Array:

   Keeps the sum of the number of edges up to and excluding the vertex for each vertex.

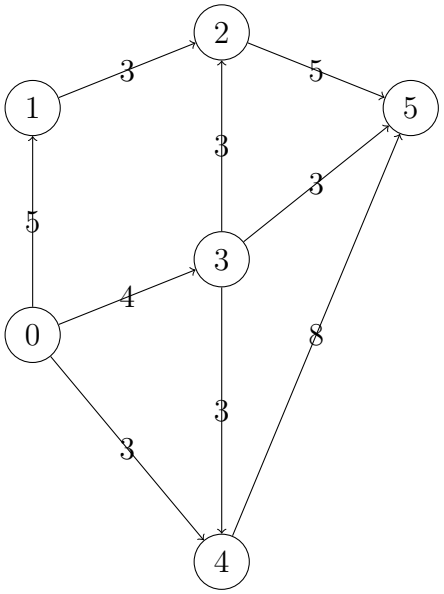
2. Column Index Array:

   Keeps the target of the edges for each vertex.

3. Weights Array:

   Keeps the weights of the edges for each vertex.

Take a graph as below:



Then, corresponding CRS arrays will be as follows:

0	**1**	**2**	**3**	**4**	**5**	**6**
0	3	4	5	8	9	9

Table 1: Row pointer array

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	3	4	2	5	2	4	5	5

Table 2: Column index array

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
5	4	3	3	5	3	3	3	8

Table 3: Weights array

Suppose you want to know all of the outgoing edges from the vertex 3. Then, first, you look at the 3^{rd} and the 4^{th} indexes of the row pointer, which are 5 and 8. Then, you look from 5^{th} to 8^{th} (8 excluded) indexes of the column index array, which are 2, 4 and 5. Similarly, 5^{th} to 8^{th} (8 excluded) indexes of the weights array will give you 3, 3 and 3. This means that there are 3 outgoing edges from 3: to 2 with weight 3, to 4 with weight 3 and to 5 with weight 3.

In this step, we have constructed the following graphs in CRS format:

1. Paper "cites" paper graph representing (citation network)
2. Paper "is cited by" paper graph (citation network)
3. Author "writes" paper graph
4. Author "collaborates with" author graph (coauthorship network)
5. Author "cites" author graph

In further steps, we have used these graphs several times. Citation networks and author "writes" paper graph have been used when evaluating papers and authors. We have also used the citation network to identify academic fields. In addition, we have used author "cites" author graph while trying to detect academic cartels.

2.1.3 Field Identification

In this step, we have identified different fields in academic network. To identify different fields, we have applied a community detection algorithm, the Louvain method [3], on paper "cites" paper graph. Then, we have analyzed the keyword and field of study frequencies to identify frequent keywords and fields of study we have previously found. Finally, we have depicted the word clouds for each of the top 5 communities and identified these communities as follows:

1. Social Sciences
2. Computer Science
3. Physics & Chemistry
4. Biology
5. Medicine

You can find the word clouds of keywords and fields of study we have used to identify these fields below:

Social Sciences

Figure 1: Fields of Study

Figure 2: Keywords

Computer Science

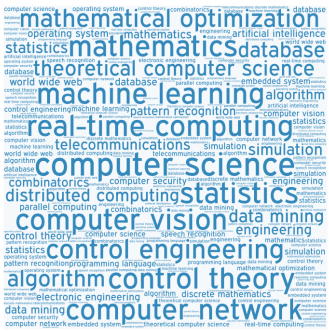


Figure 3: Fields of Study

Figure 4: Keywords

Physics & Chemistry

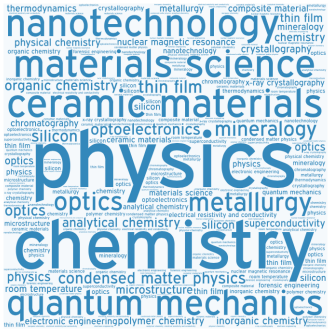


Figure 5: Fields of Study

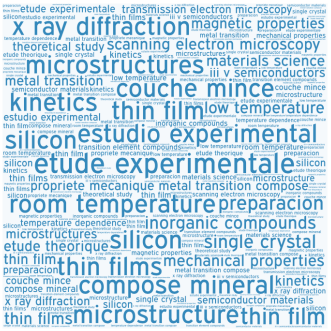


Figure 6: Keywords

Biology

Figure 7: Fields of Study

Figure 8: Keywords

Medicine

Figure 9: Fields of Study

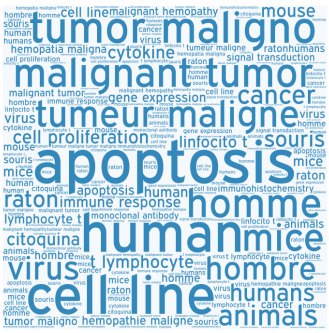


Figure 10: Keywords

2.1.4 Cartel Detection

In this step, we have tried to identify the academic cartels. For this purpose, we have applied community detection on author "cites" author graphs. In addition, we have applied community detection on author "cites" author graphs after extraction of self citations. However, we have failed to correctly identify academic cartels on both attempts.

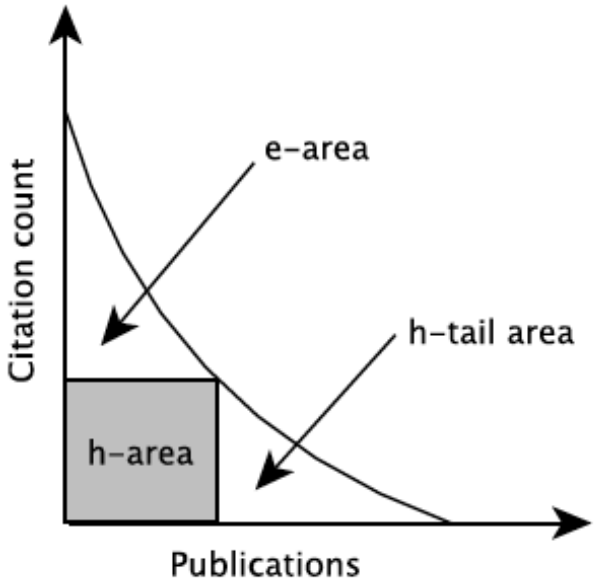
2.1.5 Isolated Community Graphs

In this step, we have isolated paper "cites" paper and paper "is cited by" paper graphs for each of the top 5 fields previously identified. During the isolation process, we have taken only the nodes belonging to the isolated community, and only the edges which are from and to a node belonging to the isolated community.

2.1.6 Existing Indexes

In this step we have evaluated the authors according to different indexes: h-index [4], g-index [5], and h'-index [6]. Moreover, we have evaluated the authors with same indexes removing self citations.

Below graph shows the distribution of papers of a researcher with respect to their citation counts:



A researcher has a h-index h if h of his papers have at least h citations and all of the remaining papers of his have less than h citations. h-index only considers the h-area of the figure above.

A researcher has a g-index g if top g papers have at least g^2 citations in total and top $g+1$ papers have less than $(g+1)^2$ citations when papers are sorted in a descending order according to thier citation counts. g-index also uses the e-area of the figure above in addition to h-area.

A researcher has an h'-index $h' = eh/t$ where h is his h-index, and e and t are the square roots of the e-area and h-tail area from the above graph, respectively. In our calculations, we have taken $h' = \frac{(e+1)}{(t+1)} \times h$ to avoid the negative effects of multiplying with and dividing to 0.

As a result of this step, we have computed h-index, g-index, and h'-index of each author both with and without self citations.

2.1.7 Comparison of Existing Indexes

In this step, we have compared h-index, g-index, and h'-index with each other. We have also investigated the effects of self citation in each index.

While comparing the indexes, we have taken the top 1 million authors with respect to these indexes, and compared the rankings.

Below graphs show the relationships between h, g and h' indexes:

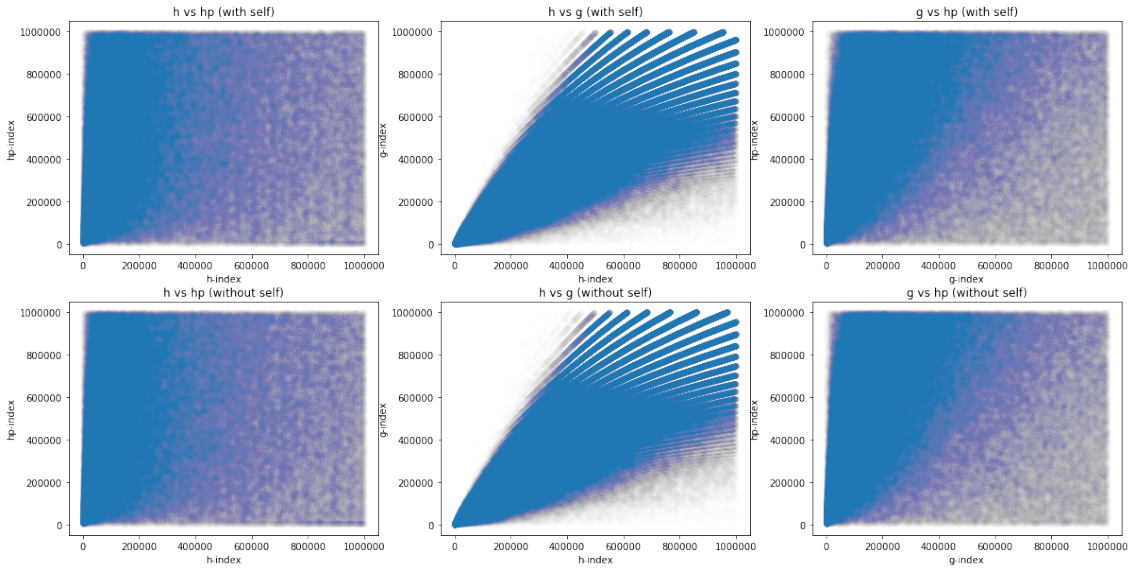


Figure 11: Comparison: h-index, g-index, h'-index

As seen in the graphs, there are no correlations between h'-index and other indexes. However, we can see a correlation between h and g indexes.

Below graphs show the effects of self citations for all of the indexes:

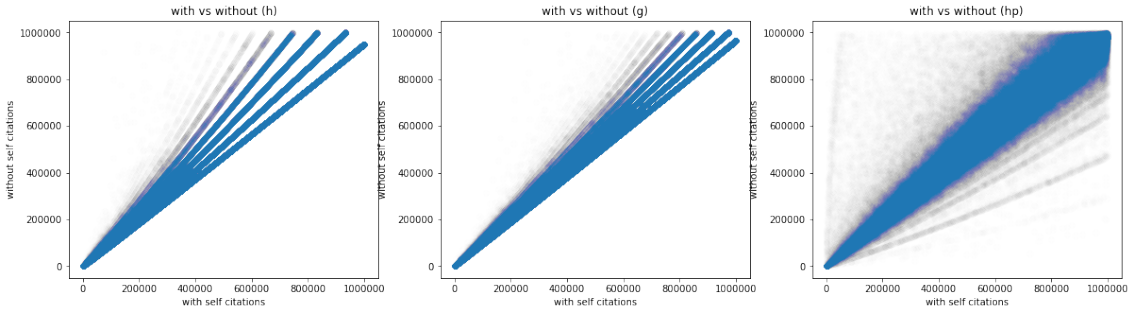


Figure 12: Comparison: with vs without self citations

As seen in the graphs, g-index is the most robust index against the effects of self citation. Then, we can count h-index as the second robust index. However, when looked at the h'-index, we see a lot of outliers.

2.1.8 Comparison of Different Fields

In this step, we have looked at the h-index in depth and compared different fields looking at h-index.

Below graph shows the h-indexes of the top 1000 authors within their communities:

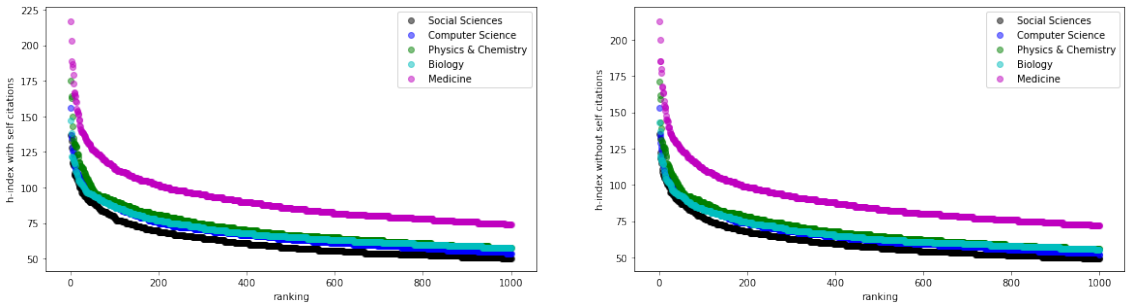


Figure 13: Top 1000 authors within top 5 community

We can see from the figure that top 1000 authors in the field of medicine has significantly higher h-indexes than other 4 fields.

Below table shows some statistics about the effects of self citation in each community:

All:	Mean: 0.028862	StDev: 0.047621	NonZeroPc: 0.372346
Social Sciences:	Mean: 0.031059	StDev: 0.050582	NonZeroPc: 0.367688
Computer Science:	Mean: 0.057076	StDev: 0.070825	NonZeroPc: 0.561095
Physics & Chemistry:	Mean: 0.044739	StDev: 0.060716	NonZeroPc: 0.484975
Biology:	Mean: 0.038590	StDev: 0.053108	NonZeroPc: 0.453741
Medicine:	Mean: 0.022223	StDev: 0.041148	NonZeroPc: 0.301951

Figure 14: Statistics about the effects of self citation

Please note that, numbers in mean and standard deviation are in scale of percentage decrease in h-index from with self citations to without self citations. For example, if an author has an h-index of 20 with self citations and 19 without self citations, he is counted as 0.05 representing a 5% decrease in h-index.

According to these statistics, the field of computer science is most prone to the effects of self citations while the field of medicine is the least prone.

2.2 Realistic Constraints

This project had various realistic constraints that can be grouped under 4 categories: (i) economic constraints, (ii) manufacturability constraints, (iii) sustainability constraints, (iv) social constraints.

Economic constaints on this project was the data we had. As a data driven project, we were tightly bound the the data we had. Firstly, although the datasets are huge, they were not complete. Secondly, the data were not perfect, and therefore needed some preprocessing.

Manufacturability constraints were the processing power we have. We have used Gandalf, a server of the school, to keep and process the data. However, although the server was enough to keep and process the data, since the data was huge, each operation has taken a lot of time, generally hours. This had a limiting effect on the speed of the project.

Sustainability constraints were based on the up-to-dateness of the data. The data we had was up to June 2017. Therefore, our results are based on the data up to that date.

Social constraints were mainly because of that we have tried to evaluate researchers and their works. Although we believe that our evaluation feflects the data we have in the most objective way, since we try to evaluate the quality, a subjective measure, the result is still subjective.

2.3 Methodology

2.3.1 Louvain Method

We have used community detection in 2 different tasks: field identification and cartel detection. In both of these tasks, we have used the Louvain method for community detection. We have used the C++ implementation of the Louvain Method by Etienne Lefebvre, who is behind the original idea of the method.

The Louvain Method is an iterative method which has 2 phases. In the beginning every node is assigned to its own community. In the first phase, for each node, the algorithm checks the neighbors of that node and if changing community will increase the modularity it changes the community. In the second phase, each community is converted into a node and by doing so, a new graph is created.

You can find the pseudocode [7] for the algorithm below:

Algorithm 1 The Louvain Method

```
Let G the initial network
while increase in modularity do
    Put each node of G in its own seperate community
    while previous modularity < new modularity do
        for all nodes do
            Calculate move for node that yields highest increase in modularity
            if there exists a move with positive gain then
                Move the mode to new community
            else
                Let the node stay in its current community
            end if
        end for
    end while
    if the new modularity is higher than the initial then
        Contract G
    end if
end while
```

2.3.2 Frequency Analysis in Field Identification

During the identification of fields, after the communities have been identified, we have analyzed the frequencies of keywords and fields of study to identify the top 5 community.

We have scored each keyword and field of study and listed them in the decreasing order of their scores. We have used 3 different scoring, which gave similar results:

1. $score = \frac{freq}{expected} \times freq$
2. $score = \sqrt{\frac{freq}{expected}} \times freq$
3. $score = \frac{freq}{expected} \times \sqrt{freq}$

where $freq$ is the frequency of a keywords within the community, and *expected* is the expected frequency obtained by the general distribution and the community size.

2.3.3 Graphs in Comparison

In the last 2 steps of our project, we have plotted different graphs for comparison purposes. To plot these graphs, we have used matplotlib library and Jupyter.

2.4 Results and Discussion

In this project, we have initially aimed to have a better understanding of the academic world, and we had planned to do it with evaluating papers, evaluating researchers and comparing fields. At the end of the project, we can comfortably say that we have done all of we had initially planned. We have evaluated papers and researchers according to different indexes, and we have compared different fields according to these indexes.

However, although we have completed all the tasks, our knowledgde about the academic world didn't change as we expected. In my opinion, we could have deepen our investigation about the academic world and find some meaningful exemplary points out of our data. I think, the data being huge and messy had badly affected us in achieving this.

2.5 Impact

In our work, we have compared the existing methods for author evaluation. Although the methods were not new, comparing existing methods are still valuable. In addition, we have evaluated the researchers according to isolated fields, which was quite unorthodox and is still open for further research.

2.6 Ethical Issues

At the beginning of the project, we had ethical concerns based on the fact that we were investigating on real people and their work. However, as stated in the discussion section, we have failed to find meaningful examples out of our data. Therefore, this concern has been dissapeared.

2.7 Project Management

Although we have performed most of our tasks, there were still some incomplete tasks according to the initial project plan. According to the original plan, we had an objective to come up with an original evaluation method based on Random Walk. However, beside of implementing PageRank [8], we have not been able to modify it. Therefore, the project has evolved from proposing original method to comparing existing methods.

2.8 Conclusion and Final Work

In this project, we have evaluated papers and researchers based on existing methods. In addition, we have identified different fields, and compared them

based on existing methods. However, the current form of the project is not final and it is up to develop. There are 2 possible developments for our project.

1. Using our methods, some exemplary points can be identified out of data. By doing so, some of our findings will be making sense.

2. Some new methods for evaluating researchers can be proposed and compared with the existing ones. By doing so, the shortcomings of existing methods can be identified better.

References

[1] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. (2008). *ArnetMiner: Extraction and Mining of Academic Social Networks.* In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining: p: 990-998.

[2] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. (2015). *An Overview of Microsoft Academic Service (MAS) and Applications.* In Proceedings of the 24th International Conference on World Wide Web: p. 243-246

[3] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, Etienne Lefebvre. (2008). *Fast Unfolding of Communities in Large Networks.* In Journal of Statistical Mechanics: Theory and Experiment 2008 (10): P10008 (12pp).

[4] Hirsch, J.E. (2005). *An index to quantify an individual's scientific research output.* In Proceedings of the National Academy of Sciences of the United States of America. 102(46): p. 16569-16572.

[5] Egghe, L. (2006). *Theory and practise of the g-index.* In Scientometrics. 69(1): p. 131-152.

[6] Zhang, C. T. (2013). *The h'-Index, Effectively Improving the h-Index Based on the Citation Distribution..* In PLoS ONE. 8(4): p. e59912.

[7] Herman Moyner Lund. (2017). *Community Detection in Complex Networks.* Master Thesis in Department of Informatics, University of Berlin.

[8] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web.* Technical report, Stanford InfoLab.

Appendix*

1. comparison-1m.pdf →

   h-index vs g-index vs h'-index – w/ vs w/o self citations

2. indepth-hindex.pdf →

   h-index in different communities

*Due to PDF export of Jupyter, there may be some defects.

comparison-1m

May 20, 2018

In [1]:
```
import matplotlib.pyplot as plt
```

In [2]:
```
hws = []
with open("top1m-h-wself.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        hws.append(int(line[:-1]))

hwos = []
with open("top1m-h-woself.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        hwos.append(int(line[:-1]))

gws = []
with open("top1m-g-wself.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        gws.append(int(line[:-1]))

gwos = []
with open("top1m-g-woself.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        gwos.append(int(line[:-1]))

hpws = []
with open("top1m-hp-wself.txt") as inp:
    inp.readline()
```

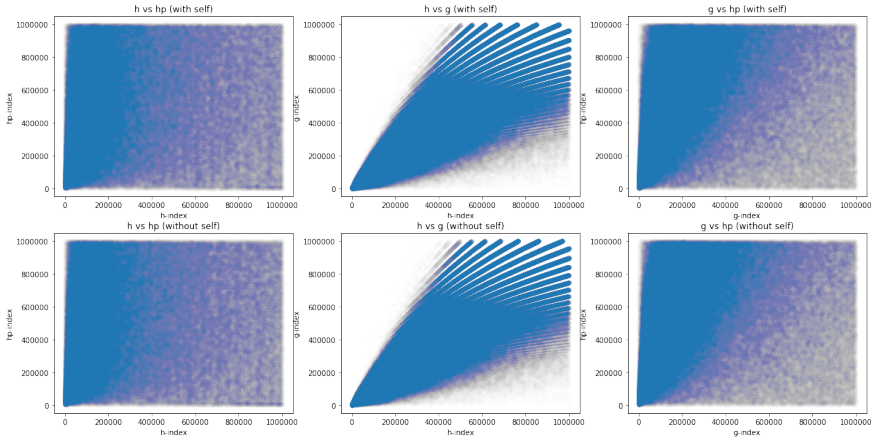
```
        inp.readline()

        for line in inp:
            hpws.append(float(line[:-1]))

    hpwos = []
    with open("top1m-hp-woself.txt") as inp:
        inp.readline()
        inp.readline()

        for line in inp:
            hpwos.append(float(line[:-1]))
```

In [3]:
```
    wself = {}
    for i, p in enumerate(hws):
        if p in wself: wself[p]["h"] = i + 1
        else: wself[p] = { "h": i + 1, "g": None, "hp": None }
    for i, p in enumerate(gws):
        if p in wself: wself[p]["g"] = i + 1
        else: wself[p] = { "g": i + 1, "h": None, "hp": None }
    for i, p in enumerate(hpws):
        if p in wself: wself[p]["hp"] = i + 1
        else: wself[p] = { "hp": i + 1, "h": None, "g": None }

    woself = {}
    for i, p in enumerate(hwos):
        if p in woself: woself[p]["h"] = i + 1
        else: woself[p] = { "h": i + 1, "g": None, "hp": None }
    for i, p in enumerate(gwos):
        if p in woself: woself[p]["g"] = i + 1
        else: woself[p] = { "g": i + 1, "h": None, "hp": None }
    for i, p in enumerate(hpwos):
        if p in woself: woself[p]["hp"] = i + 1
        else: woself[p] = { "hp": i + 1, "h": None, "g": None }
```

In [4]:
```
    plt.figure(figsize=(20, 10))

    hwshp = []
    hpwsh = []
    for p in wself:
        if wself[p]["h"] is not None and wself[p]["hp"] is not None:
            hwshp.append(wself[p]["h"])
            hpwsh.append(wself[p]["hp"])

    plt.subplot(2, 3, 1)
    plt.scatter(hwshp, hpwsh, alpha=0.002)
    plt.xlabel("h-index")
    plt.ylabel("hp-index")
```

```
plt.title("h vs hp (with self)")

hwsg = []
gwsh = []
for p in wself:
    if wself[p]["h"] is not None and wself[p]["g"] is not None:
        hwsg.append(wself[p]["h"])
        gwsh.append(wself[p]["g"])

plt.subplot(2, 3, 2)
plt.scatter(hwsg, gwsh, alpha=0.002)
plt.xlabel("h-index")
plt.ylabel("g-index")
plt.title("h vs g (with self)")

gwshp = []
hpwsg = []
for p in wself:
    if wself[p]["g"] is not None and wself[p]["hp"] is not None:
        gwshp.append(wself[p]["g"])
        hpwsg.append(wself[p]["hp"])

plt.subplot(2, 3, 3)
plt.scatter(gwshp, hpwsg, alpha=0.002)
plt.xlabel("g-index")
plt.ylabel("hp-index")
plt.title("g vs hp (with self)")

hwoshp = []
hpwosh = []
for p in woself:
    if woself[p]["h"] is not None and woself[p]["hp"] is not None:
        hwoshp.append(woself[p]["h"])
        hpwosh.append(woself[p]["hp"])

plt.subplot(2, 3, 4)
plt.scatter(hwoshp, hpwosh, alpha=0.002)
plt.xlabel("h-index")
plt.ylabel("hp-index")
plt.title("h vs hp (without self)")

hwosg = []
gwosh = []
for p in woself:
    if woself[p]["h"] is not None and woself[p]["g"] is not None:
        hwosg.append(woself[p]["h"])
        gwosh.append(woself[p]["g"])
```

```
plt.subplot(2, 3, 5)
plt.scatter(hwosg, gwosh, alpha=0.002)
plt.xlabel("h-index")
plt.ylabel("g-index")
plt.title("h vs g (without self)")

gwoshp = []
hpwosg = []
for p in woself:
    if woself[p]["g"] is not None and woself[p]["hp"] is not None:
        gwoshp.append(woself[p]["g"])
        hpwosg.append(woself[p]["hp"])

plt.subplot(2, 3, 6)
plt.scatter(gwoshp, hpwosg, alpha=0.002)
plt.xlabel("g-index")
plt.ylabel("hp-index")
plt.title("g vs hp (without self)")

plt.show()
```

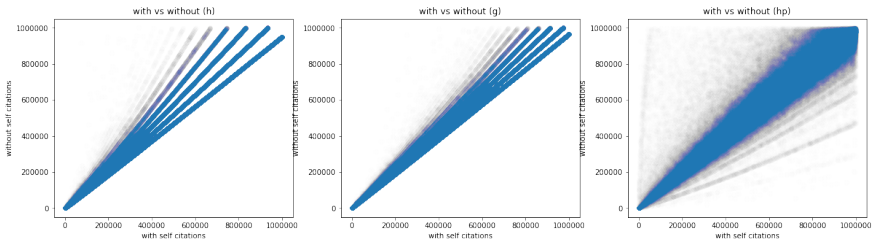


In [5]:
```
hwwo = {}
for i, p in enumerate(hws):
    if p in hwwo: hwwo[p]["w"] = i + 1
    else: hwwo[p] = { "w": i + 1, "wo": None }
for i, p in enumerate(hwos):
    if p in hwwo: hwwo[p]["wo"] = i + 1
    else: hwwo[p] = { "wo": i + 1, "w": None }
```

```
gwwo = {}
for i, p in enumerate(gws):
    if p in gwwo: gwwo[p]["w"] = i + 1
    else: gwwo[p] = { "w": i + 1, "wo": None }
for i, p in enumerate(gwos):
    if p in gwwo: gwwo[p]["wo"] = i + 1
    else: gwwo[p] = { "wo": i + 1, "w": None }

hpwwo = {}
for i, p in enumerate(hpws):
    if p in hpwwo: hpwwo[p]["w"] = i + 1
    else: hpwwo[p] = { "w": i + 1, "wo": None }
for i, p in enumerate(hpwos):
    if p in hpwwo: hpwwo[p]["wo"] = i + 1
    else: hpwwo[p] = { "wo": i + 1, "w": None }
```

In [6]:
```
plt.figure(figsize=(20, 5))

hwvswo = []
hwovsw = []
for p in hwwo:
    if hwwo[p]["w"] is not None and hwwo[p]["wo"] is not None:
        hwvswo.append(hwwo[p]["w"])
        hwovsw.append(hwwo[p]["wo"])

plt.subplot(1, 3, 1)
plt.scatter(hwvswo, hwovsw, alpha=0.002)
plt.xlabel("with self citations")
plt.ylabel("without self citations")
plt.title("with vs without (h)")

gwvswo = []
gwovsw = []
for p in gwwo:
    if gwwo[p]["w"] is not None and gwwo[p]["wo"] is not None:
        gwvswo.append(gwwo[p]["w"])
        gwovsw.append(gwwo[p]["wo"])

plt.subplot(1, 3, 2)
plt.scatter(gwvswo, gwovsw, alpha=0.002)
plt.xlabel("with self citations")
plt.ylabel("without self citations")
plt.title("with vs without (g)")

hpwvswo = []
hpwovsw = []
for p in hpwwo:
    if hpwwo[p]["w"] is not None and hpwwo[p]["wo"] is not None:
```

```
            hpwvswo.append(hpwwo[p]["w"])
            hpwovsw.append(hpwwo[p]["wo"])

    plt.subplot(1, 3, 3)
    plt.scatter(hpwvswo, hpwovsw, alpha=0.002)
    plt.xlabel("with self citations")
    plt.ylabel("without self citations")
    plt.title("with vs without (hp)")

    plt.show()
```



In [ ]:

indepth-hindex

May 20, 2018

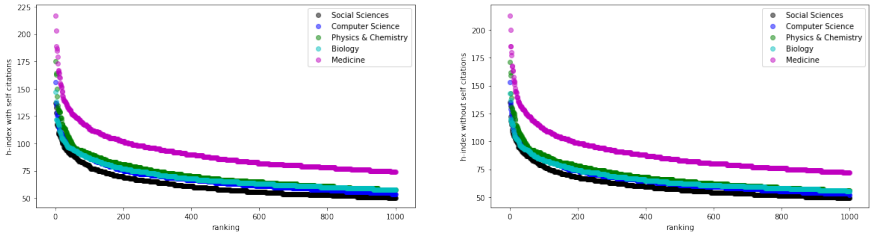
In [1]:
```
import numpy as np
import scipy.stats
import matplotlib.pyplot as plt
```

In [2]:
```
hws_all, hwos_all, ind_all = [], [], []
with open("indepth-all.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        ind_all.append(int(line[:-1].split(' ')[0]))
        hws_all.append(int(line[:-1].split(' ')[1]))
        hwos_all.append(int(line[:-1].split(' ')[2]))

hws_1, hwos_1, ind_1 = [], [], []
with open("indepth-1.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        ind_1.append(int(line[:-1].split(' ')[0]))
        hws_1.append(int(line[:-1].split(' ')[1]))
        hwos_1.append(int(line[:-1].split(' ')[2]))

hws_2, hwos_2, ind_2 = [], [], []
with open("indepth-2.txt") as inp:
    inp.readline()
    inp.readline()

    for line in inp:
        ind_2.append(int(line[:-1].split(' ')[0]))
        hws_2.append(int(line[:-1].split(' ')[1]))
        hwos_2.append(int(line[:-1].split(' ')[2]))

hws_3, hwos_3, ind_3 = [], [], []
with open("indepth-3.txt") as inp:
    inp.readline()
```

```
        inp.readline()

        for line in inp:
            ind_3.append(int(line[:-1].split(' ')[0]))
            hws_3.append(int(line[:-1].split(' ')[1]))
            hwos_3.append(int(line[:-1].split(' ')[2]))

    hws_4, hwos_4, ind_4 = [], [], []
    with open("indepth-4.txt") as inp:
        inp.readline()
        inp.readline()

        for line in inp:
            ind_4.append(int(line[:-1].split(' ')[0]))
            hws_4.append(int(line[:-1].split(' ')[1]))
            hwos_4.append(int(line[:-1].split(' ')[2]))

    hws_5, hwos_5, ind_5 = [], [], []
    with open("indepth-5.txt") as inp:
        inp.readline()
        inp.readline()

        for line in inp:
            ind_5.append(int(line[:-1].split(' ')[0]))
            hws_5.append(int(line[:-1].split(' ')[1]))
            hwos_5.append(int(line[:-1].split(' ')[2]))
```

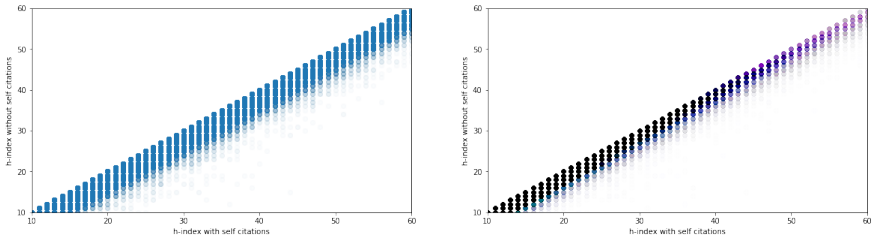
In [3]:
```
    hws_max_1, hwos_max_1 = sorted(hws_1, reverse=True)[:1000], sorted(hwos_1,
    hws_max_2, hwos_max_2 = sorted(hws_2, reverse=True)[:1000], sorted(hwos_2,
    hws_max_3, hwos_max_3 = sorted(hws_3, reverse=True)[:1000], sorted(hwos_3,
    hws_max_4, hwos_max_4 = sorted(hws_4, reverse=True)[:1000], sorted(hwos_4,
    hws_max_5, hwos_max_5 = sorted(hws_5, reverse=True)[:1000], sorted(hwos_5,

    plt.figure(figsize=(20, 5))

    plt.title("h-index of top 1000")

    ax1 = plt.subplot(1, 2, 1)
    ax1.set_xlabel("ranking")
    ax1.set_ylabel("h-index with self citations")

    ax1.scatter(np.arange(1000) + 1, hws_max_1, c="k", alpha=0.5, label="Social
    ax1.scatter(np.arange(1000) + 1, hws_max_2, c="b", alpha=0.5, label="Comput
    ax1.scatter(np.arange(1000) + 1, hws_max_3, c="g", alpha=0.5, label="Physic
    ax1.scatter(np.arange(1000) + 1, hws_max_4, c="c", alpha=0.5, label="Biolog
    ax1.scatter(np.arange(1000) + 1, hws_max_5, c="m", alpha=0.5, label="Medici

    ax1.legend(loc='upper right')
```

```
ax2 = plt.subplot(1, 2, 2)
ax2.set_xlabel("ranking")
ax2.set_ylabel("h-index without self citations")

ax2.scatter(np.arange(1000) + 1, hwos_max_1, c="k", alpha=0.5, label="Socia
ax2.scatter(np.arange(1000) + 1, hwos_max_2, c="b", alpha=0.5, label="Compu
ax2.scatter(np.arange(1000) + 1, hwos_max_3, c="g", alpha=0.5, label="Physi
ax2.scatter(np.arange(1000) + 1, hwos_max_4, c="c", alpha=0.5, label="Biolo
ax2.scatter(np.arange(1000) + 1, hwos_max_5, c="m", alpha=0.5, label="Medic

ax2.legend(loc='upper right')

plt.show()
```



In [4]:
```
plt.figure(figsize=(20, 5))

plt.title("h-index -- with vs without self citations")

ax1 = plt.subplot(1, 2, 1)
ax1.set_xlabel("h-index with self citations")
ax1.set_ylabel("h-index without self citations")

ax1.scatter(hws_all, hwos_all, alpha=0.02)

ax1.set_xlim(10, 60)
ax1.set_ylim(10, 60)

ax2 = plt.subplot(1, 2, 2)
ax2.set_xlabel("h-index with self citations")
ax2.set_ylabel("h-index without self citations")

ax2.scatter(hws_1, hwos_1, c="k", alpha=0.005)
ax2.scatter(hws_2, hwos_2, c="b", alpha=0.005)
ax2.scatter(hws_3, hwos_3, c="g", alpha=0.005)
ax2.scatter(hws_4, hwos_4, c="c", alpha=0.005)
```

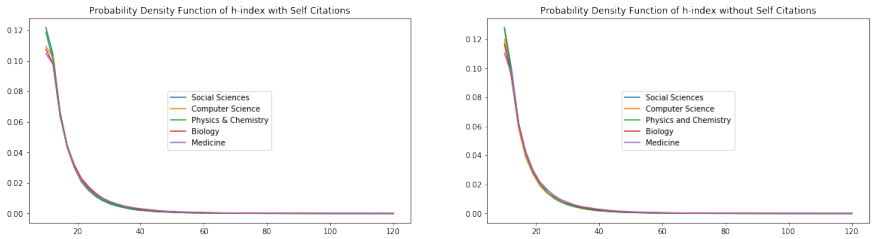
```
        ax2.scatter(hws_5, hwos_5, c="m", alpha=0.005)

        ax2.set_xlim(10, 60)
        ax2.set_ylim(10, 60)

        plt.show()
```




```
In [6]: hdiff_all = [(hws_all[i] - hwos_all[i]) / hws_all[i] for i in range(len(hws

        hdiff_1 = [(hws_1[i] - hwos_1[i]) / hws_1[i] for i in range(len(hws_1))]
        hdiff_2 = [(hws_2[i] - hwos_2[i]) / hws_2[i] for i in range(len(hws_2))]
        hdiff_3 = [(hws_3[i] - hwos_3[i]) / hws_3[i] for i in range(len(hws_3))]
        hdiff_4 = [(hws_4[i] - hwos_4[i]) / hws_4[i] for i in range(len(hws_4))]
        hdiff_5 = [(hws_5[i] - hwos_5[i]) / hws_5[i] for i in range(len(hws_5))]

        print("All:\t\t\t\t\t\tMean: {:.6f}\t\tStDev: {:.6f}\t\tNonZeroPc: {:8.6f}"
              .format(np.mean(hdiff_all), np.std(hdiff_all), (len(hdiff_all) - hdif

        print("Social Sciences:\t\t\tMean: {:.6f}\t\tStDev: {:.6f}\t\tNonZeroPc:
              .format(np.mean(hdiff_1), np.std(hdiff_1), (len(hdiff_1) - hdiff_1.co
        print("Computer Science:\t\tMean: {:.6f}\t\tStDev: {:.6f}\t\tNonZeroPc:
              .format(np.mean(hdiff_2), np.std(hdiff_2), (len(hdiff_2) - hdiff_2.co
        print("Physics & Chemistry:\t\tMean: {:.6f}\t\tStDev: {:.6f}\t\tNonZero
              .format(np.mean(hdiff_3), np.std(hdiff_3), (len(hdiff_3) - hdiff_3.co
        print("Biology:\t\t\t\tMean: {:.6f}\t\tStDev: {:.6f}\t\tNonZeroPc: {:8.6f
              .format(np.mean(hdiff_4), np.std(hdiff_4), (len(hdiff_4) - hdiff_4.co
        print("Medicine:\t\t\t\tMean: {:.6f}\t\tStDev: {:.6f}\t\tNonZeroPc: {:8.6
              .format(np.mean(hdiff_5), np.std(hdiff_5), (len(hdiff_5) - hdiff_5.co
```

```
All:                                    Mean: 0.028862              S
Social Sciences:                    Mean: 0.031059              StDev
Computer Science:                    Mean: 0.057076              StDe
Physics & Chemistry:                    Mean: 0.044739              S
Biology:                            Mean: 0.038590              StDev
Medicine:                            Mean: 0.022223              StDe
```

In [7]:
```
plt.figure(figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.title("Probability Density Function of h-index with Self Citations")

kde = scipy.stats.gaussian_kde(hws_1)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Social Sci
kde = scipy.stats.gaussian_kde(hws_2)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Computer S
kde = scipy.stats.gaussian_kde(hws_3)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Physics &
kde = scipy.stats.gaussian_kde(hws_4)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Biology')
kde = scipy.stats.gaussian_kde(hws_5)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Medicine')

plt.legend(loc='center')

plt.subplot(1, 2, 2)
plt.title("Probability Density Function of h-index without Self Citations")

kde = scipy.stats.gaussian_kde(hwos_1)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Social Sci
kde = scipy.stats.gaussian_kde(hwos_2)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Computer S
kde = scipy.stats.gaussian_kde(hwos_3)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Physics ar
kde = scipy.stats.gaussian_kde(hwos_4)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Biology')
kde = scipy.stats.gaussian_kde(hwos_5)
plt.plot(np.linspace(10, 120), kde(np.linspace(10, 120)), label='Medicine')

plt.legend(loc='center')

plt.show()
```



In [ ]: