Matching Algorithm

Kalia Barkai

Minerva Schools at KGI
San Francisco, CA 94103
kalia@minerva.kgi.edu

August 2017

The matching algorithm optimises matches between student mentors and admitted students based on country of origin, secondary status (high school, gap year or transfer student) and interests, while distributing the admitted students so that each mentor has a manageable load. The algorithm is coded in Python and the data extraction and analysis is conducted through the Pandas library. The matching algorithm is a more efficient and informed way to pair mentors and admitted students, compared to the current system which is manually done on a spreadsheet and does not consider many factors for matching. However, the success of the algorithm can only be measured once it has been used. This can be done through a survey in which students rate the success of their match, and this will be compared to the score given by the matching algorithm.

1 Introduction

When deciding which university to attend, an admitted student (hereon referred to as an "admit") compares whether their background, interests and goals align with those of the university. Therefore, the opportunity to speak to a current student (mentor) gives the admit the ability to inquire into the experience of that student. This involves the mentor relaying information about how and why they made their decision and how the experience has lived up to their expectations.

A few similarities between the admit and mentor could make their interaction more effective. A similar background means that there is some common ground between the students and they might have had similar deliberations about leaving their home country, for example. Their status while applying (as a high school, gap year, or transfer student) can help make sure that

the mentor understands what considerations the student has; are they leaving home for the first time, are they starting university older than most or are they making the decision to leave their current institution and attend this one? Lastly, admits look for an institution that can offer them the resources to extend their interests and pursue their goals. Therefore, a mentor whose interests align with those of the admit, will have greater ability to encourage or discourage the admit from attending the university based on those interests.

2 Background

Currently, Minerva matches mentors and admits mainly based on country of origin or region and makes sure to distribute the admits between the mentors in that region. In some smaller admit batches, such as Binding Decision[1] and Rolling Decision[2], interests are taken into consideration when matching, and this is done subjectively by the outreach manager by looking at the interests of the admit from their accomplishments[3] and either from personal experience with the mentor, or through the interests of the mentor via Minerva's annual student interest form (PI-Q[4]).

In terms of the information available for this algorithm, the application to Minerva is the only form of data we have about the admit, while we have more structured data about the mentors through databases already built about their country of origin and outreach region and the PI-Q. It will also be easier to survey mentors for any missing data, such as their secondary status when applying to Minerva. As part of the application, the admit has to fill out their country of origin and their secondary status, while the only way to extract their interests is through their accomplishments. During application processing the accomplishments are categorised. These categories can then be used as an estimate of the admit's interests.

[1]When applying for Minerva, if an applicant is sure they want to attend Minerva they can choose to confirm their enrollment, if accepted, and in return get notified about their decision within three weeks of completing the application.

[2]These are students who apply to Minerva after Regular Decision and therefore they are notified of their admittance decision as soon as their application has been reviewed to give them sufficient time to make their enrollment decision.

[3]Applicants have to provide one or more descriptions of their accomplishments as part of their Minerva application

[4]Personal Interests Questionnaire

3 Approach and Experiments

3.1 Data Extraction

The format of the data used for the algorithm is an Excel document with two sheets: “mentors” and “admits”. “mentors” has 5 columns of data: “MENTOR”, “COUNTRY”, “REGION”, “2-STATUS” and “INTERESTS”, where “REGION” is the Outreach region of that mentor. “2-STATUS” refers to the secondary status of the mentor when they were applying to Minerva and can be “High School”, “Transfer” or “Gap Year”. “admits” has 4 columns of data: “ADMIT”, “COUNTRY”, “2-STATUS” and “INTERESTS”. For both admits and mentors the interests are each one word with a space between them in the Excel cell.

Minerva has 5 different regions: Latin America, North America, Africa, Europe and Asia. Each mentor is assigned a region, usually based on their country of origin and aligned with their Minerva Outreach Team position. This is then later used in the algorithm to check whether the admit’s country falls into the mentor’s Outreach region.

The extraction of admit and mentor interests involves three major steps: extracting admit accomplishment categories, mentor interests from PI-Q, and then converting those categories and interests into broader common interests.

The accomplishments are categorised by application processors. There are 50 different categories. Each accomplishment is labeled by a Minerva ID given to each applicant, and therefore a list of categories is made for each admit by gathering the category label for each of their accomplishments. The list of interests per admit therefore ranges from 1 to 5 which are the minimum and maximum amount of accomplishments an applicant can submit, respectively.

There are 34 different interests in the PI-Q survey for current Minerva students. Using the mentor name, the list of their interests is extracted by their answers to the survey. Here, the data shows an average of 7 interests per student.

The final list of interests includes 17 different broader topics. These new topics encompass all the interests from the accomplishment categories and from the PI-Q, but generalises them so that they can fit under the same labels. From this, list comprehension is used where any interest from the admit or mentor which falls into a specific list of interests is replaced with the broader topic and duplicates are deleted. The average amount of interests per mentor and admit is rounded up to 5 and 3, respectively.

3.2 Algorithm, Matching and Testing

3.2.1 Algorithm

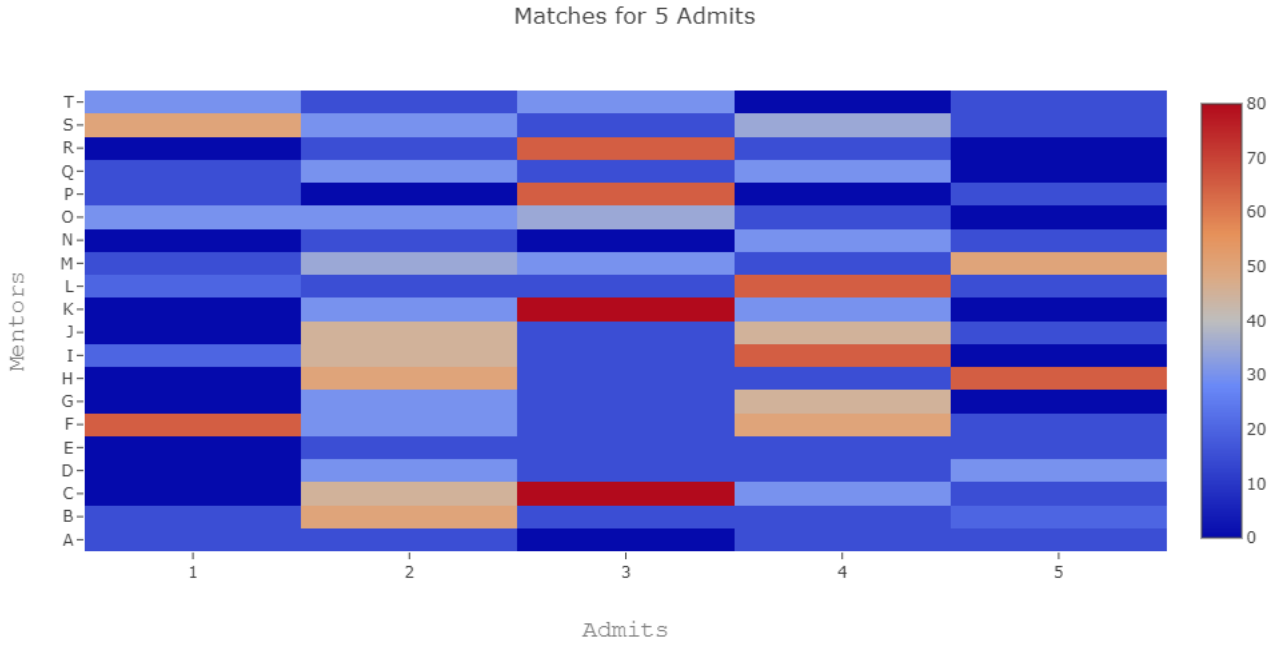
The algorithm is based on a set of lists for mentors (name, country, region, secondary status and interests) and admits (name, country, secondary status and interests) which are ordered so that the same location of an item in each list refers to the same admit's or mentor's data.

The algorithm then proceeds to check for equalities between country, country within mentor region, secondary status and interests for each mentor and admit pair. If a mentor and admit have the same country of origin 50 "points" is added to the match score, if the country of origin is not the same, but the admit's country falls into the mentor's region, then 20 points are added. An equality in secondary status and for each interest in common rewards the match another 15 points. In this way, the algorithm greatly favours matches with mentors and admits from the same country of origin and if there is more than one mentor with the same country of origin, then secondary status and interests will specify which match might be more appropriate. The algorithm then returns a list of matches and their scores. The code snippet is shown in Listing 1. below:

```
1  def match(iden, country, status, interests):
2    x = 0
3      while x < len(mentors):
4        score = 0
5          mentor = mentors.iloc[x]
6          admit = iden[y]
7          if country[y][0] in mentor_country[x]:
8            score += 50
9          # the following only adds 20 points if admit country does
       not equal mentor country
10         if country[7][0] not in mentor_country[x] and country[y
       ][0] in mentor_region[x]:
11           score += 20
12         if mentor_status[x] == status[y]:
13           score += 15
14         for interest in interests[y]:
```

```
            if interest in mentor_interests[x]:
                score += 15
        applicant.append(admit)
        student.append([mentor[0]])
        scores.append(score)
        y += 1
    x += 1
```

Listing 1: Matching algorithm function which takes in four lists: the ID (iden) country of origin (country) secondary status (status) and interests (interests) of admits. The lists; 'applicant' 'student' and 'scores' are defined globally outside of the function.



Heatmap 1. Heatmap of matching scores for each mentor (y-axis) and admit (x-axis) when the algorithm takes in 5 admits and 20 different mentors. Heatmap made using Plotly library on Python. High scoring matches in the 60s are due to same country of origin between mentor and admits. Some admits have more than one top mentor match and this is handled by deleting duplicates to optimise distribution, as described in section 3.2.2 below.

3.2.2 Matching

After all the matching scores were assigned, using Pandas .groupby() and .transform(max) functions, the data was grouped by the matches per admit, and then only the max score per admit was saved into a new dataframe. This can be seen in Listing 2.

```
by_admit = matches.groupby("ADMIT")["SCORE"].transform(max) ==
    matches["SCORE"]
top_matches = matches[by_admit]
```

Listing 2: "matches" is the dataframe of matches and scores. This code keeps only the top scores of each admit.

Since an admit can have the same top score match with more than one mentor, it is necessary to delete matches so that each admit is left with only one mentor. Using Pandas this can be done in two ways, by either keeping the first or last duplicate score for the admit. Therefore, in order to create maximum distribution of admits assigned to mentors, the mentors with less admit matches should keep the admit for the duplicates. To do this, we group the dataframe by mentor and then add an extra column to the dataframe called "count" which counts the number of rows that mentor appears and therefore the number of matches that mentor has. We then use this column to order the dataframe in descending order from the mentor with the least amount of matches to the mentor with the most matches. Now, when dealing with duplicates we use Pandas to keep the first duplicate only, which corresponds always to the mentor with the least amount of matches. This is done in Listing 3.

```
# dataframe which sorts based on the mentor with the least to
    most matches
top_matches["count"] = top_matches.groupby(["MENTOR"])["APPLICANT
    "].transform("count")
top_matches = top_matches.sort_values("count")

# drop duplicates, keeping only the first match (which is the
    mentor with the least matches)
top_matches_only = top_matches.drop_duplicates("APPLICANT", keep
    = "first")
```

```
del top_matches_only["count"] #deleting old count column
```

Listing 3: Code which sorts the dataframe based on the number of matches a mentor has and then keeps only the first duplicate in order to optimise distribution.

3.2.3 Testing

Since Minerva does not admit students all in one batch, it is necessary for the algorithm to work for multiple admit cycles. This is done in two ways. In Listing 1. of the algorithm, the function uses global mentor data, but it takes in admit data, and therefore the same mentors can be used for different admit batches. Secondly, in order to still optimise distribution, the algorithm needs to take into account its previous matches, without changing them. To do this, we add an extra line of code after it keeps only the top scores of the current matching cycle and before sorting the dataframe by the mentors with the least amount of matches. This can be seen in Listing 4. This code concatenates the previous matches with the current matches.

```
top_matches = pd.concat([final_matches, top_matches], axis = 0)
```

Listing 4: Code which uses Pandas to concatenate two datasets with the same column headings where "final_matches" is the dataset of previous matches and "top_matches" is the dataset of current matches made.

To check what was the effect of matching during cycles versus all the admits at once, we created 8 fake admit batches which consisted of 3 sets of 5 students who are admitted before Early Action[5] through Binding Decision, 100 students admitted during Early Action, 300 students admitted during Regular Decision and 3 sets of 15 students admitted during Rolling Decision. After this, we compared the standard deviation of the amount of matches per mentor from the cycled-matching and straight-matching final dataframe. The cycled-matching had a standard deviation of 10.07 while the straight-matching had a standard deviation 10.90. This means that there is slightly better distribution in the cycled-matching process. In both cases, only the top scores per admit are kept, and therefore there should be no difference in the amount of high scoring matches between cycled- or straight-matching.

3.2.4 Brief Analysis

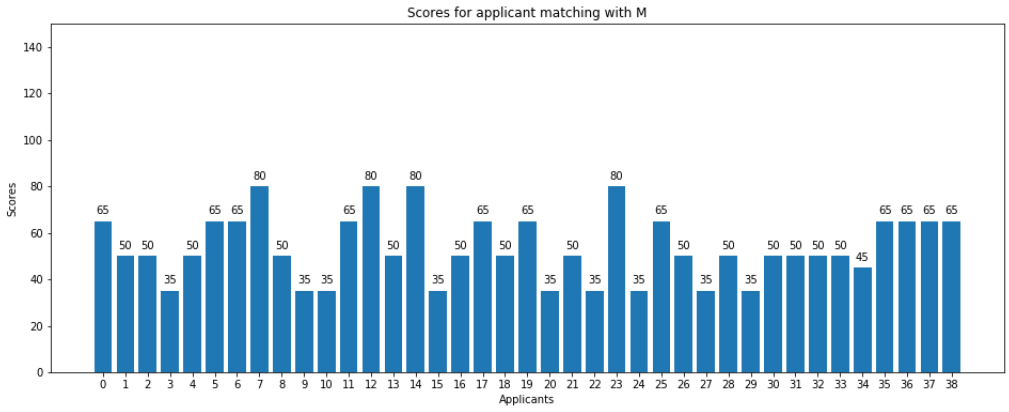
During straight-matching there is only one opportunity to order the mentors by the number of matches and when duplicates are deleted the reordering is not changed. While in cycled-

[5]Early Action consists of students who apply for Minerva between August and November of the year before enrollment, and are notified of their decision in December

matching, the order of mentors in the dataframe is updated during every cycle meaning that it adjusts based on previous matches before making new matches. Therefore, the more batches of admits and the less admits in those batches would lead to the better distribution.

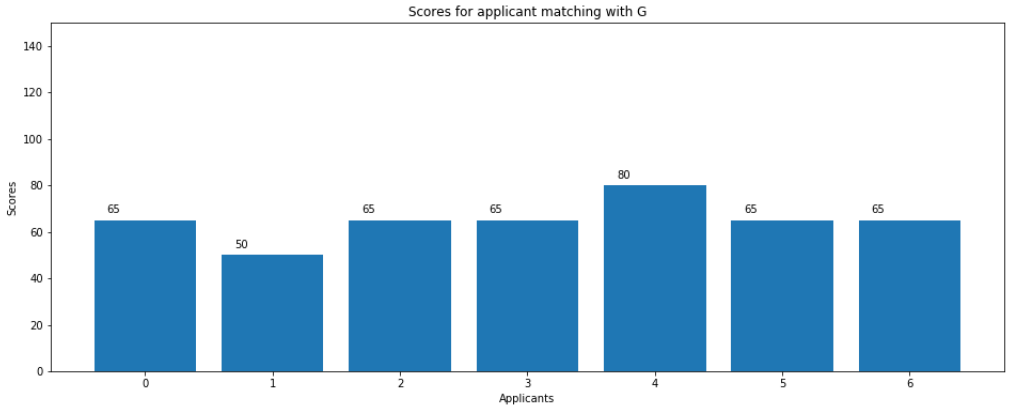
In terms of distribution, there are a number of factors which need to be considered. However, the issues arising from these factors are difficult to mitigate when testing with fake data. Firstly, some of the mentors had more than one country of origin because they would feel most comfortable to be paired with students from those countries within their region. However, since this is not consistent throughout all the mentors, mentors who have more countries listed will be assigned more students because of the high scoring of country of origin. Similarly, the more interests a mentor has, the greater possibility of that mentor receiving high score matches. On top of that, some interests are much more common than others amongst admits. For example, "Psychology" is only found as an interest for 3 admits, while "STEM Research" applies to 120 admits. There are also many more high school students from the admits than transfer and gap year students, but this is also the case with the mentors as students with a high school secondary status make up 55% of the mentors.

Bar Graph 1. and Bar Graph 2. show the matches and their scores for mentors M and G which are the mentors with the most and least scores, respectively.



Bar Graph 1. Bar graph showing the 39 matches for mentor M and the scores for each match

Table 1. Compares the entries for Mentor M and mentor G.



Bar Graph 2. Bar graph showing the 7 matches for mentor G and the scores for each match

	Mentor M	**Mentor G**
Country of Origin	Kenya	Argentina
Region	Africa	Latin America
Secondary Status	High School	High School
Interests	leadership, athletics, psychology, service	public speaking, journalism and media, service, politics, foreign language

Table 1. Table showing the different entries for Mentor M and Mentor G.

In terms of the admit data[6], there are 57 admits from Latin America, but only 2 from Argentina, while there are 68 admits from Africa and 17 from Kenya. Along with this, there are 3 mentors in the Africa region with 4 mentors assigned to Latin America. From this, we can deduce that the reason for the disparity between Mentor M and Mentor G is more due to the distribution of admits from their countries and the amount of mentors assigned to the region, than the secondary status and interests of the mentors. Therefore, it would be more useful to compare two mentors within the same region. We can do this by comparing Mentor J with Mentor G in Table 2. Mentor J had a total of 25 matches in comparison to Mentor G's 7 matches.

[6]The admit data used is not the actual data of current admits.

	Mentor J	**Mentor G**
Country of Origin	Brazil	Argentina
Region	Latin America	Latin America
Secondary Status	High School	High School
Interests	computer/technology, education, journalism and media, business	public speaking, journalism and media, service, politics, foreign language

Table 2. Table showing the different entries for Mentor J and Mentor G.

In this case, it is important to note that of the 57 admits from Latin America, 40 were from Brazil, explaining the difference in amount of matches between Mentor J and Mentor G.

3.3 Web-App

Lastly, to make the algorithm more accessible and easier to use, a simple web-application was created using Flask and hosted on pythonanywhere.com. The user uploads an Excel sheet with the headers as described in 3.1 Data Extraction and the application returns an Excel sheet of the matches, while simultaneously storing the matches. This allows the user to then use the same application for the next batch and it will use the stored matches to distribute the new matches. It will then return an Excel sheet of all the matches, new and old.

4 Conclusion and Suggestions

Using a matching algorithm has benefits over the current system as it considers more than just country of origin, is more efficient and can be adjusted easily based on specific feedback. However, since the weighting of country of origin is so high it means that in the majority of cases, if there is the same country of origin, the match will be made. Therefore, it might be beneficial to play around with the weightings of each similarity depending on the importance of distribution versus similar backgrounds. Other metrics can be added here such as same home language which fills some of the roles as same country of origin.

Another way to increase distribution is to have more controlled mentor data. This means to make sure that mentors have an equal number of interests, and possibly even distributing the more popular interest topics between the mentors, if appropriate. Secondly, if mentors would like to have a preference for countries which are not their country of origin, then I would suggest

adding an extra column to the initial spreadsheet with these countries. This way, the algorithm can be updated to give more points if the admit is from one of these countries, but not as many points as if they were from the same country.

Depending on the number of admits, I would also suggest that during the big admit batches of Early Action and Regular Decision, the spreadsheets are divided and uploaded to the application in parts. This way allowing for the number of matches per mentor to be reordered more often. To make this easier, the code could also be updated, that if the spreadsheet exceeds a certain amount, then the algorithm divides it itself and only takes in a certain number of admits during each matching process.

Lastly, the only way to measure the success of the matching algorithm is to survey the matches after the mentor/admit season. It would be useful in this survey to sample low scoring and high scoring matches and compare the opinions of the admits and mentors, as well as the amount of final enrollees from these groups.

References

Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. **The NumPy Array: A Structure for Efficient Numerical Computation**, Computing in Science & Engineering, **13**, 22-30 (2011), DOI:10.1109/MCSE.2011.37

John D. Hunter.**Matplotlib: A 2D Graphics Environment**, Computing in Science & Engineering, **9**, 90-95 (2007), DOI:10.1109/MCSE.2007.55

Wes McKinney. **Data Structures for Statistical Computing in Python**, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

Plotly Technologies Inc. Collaborative data science. Montréal, QC, 2015. https://plot.ly.