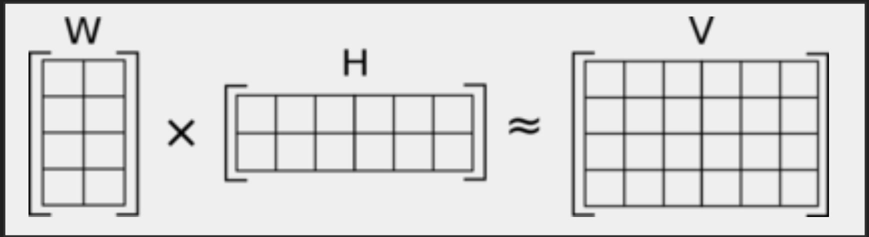
# Non-Negative Matrix Factorization

NNMF

# What is NMF?

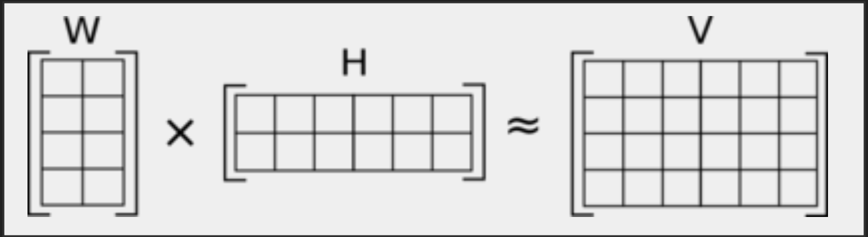
- Non-Negative Matrix Factorization (NMF) - For a matrix V of dimension *m x n* where each element $V_{ij} \geq 0$, the matrix is decomposed into two matrices W and H of dimensions *m x k* and *k x n*, respectively, where each element $w_{ij} \geq 0$ and $h_{ij} \geq 0$ and k < min(*m*, *n*) such that:

$$W \times H = V$$



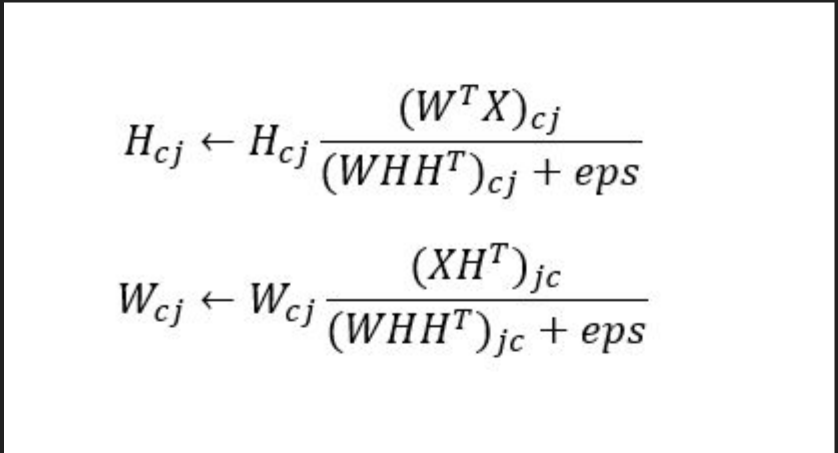
# What is NMF cont...

- V is decomposed into a long matrix W and wide matrix H.
- *k* is specified by the user as long as *k* < min (*m, n*). This specifies the number of clusters the algorithm will create
- Each column of V, or $v_i$, can be calculated as: = W × $h_i$



# What is NMF cont...

- Lee and Seung's multiplicative update rule



$$H_{cj} \leftarrow H_{cj} \frac{(W^T X)_{cj}}{(WHH^T)_{cj} + eps}$$

$$W_{cj} \leftarrow W_{cj} \frac{(XH^T)_{jc}}{(WHH^T)_{jc} + eps}$$

# Why we use NMF?

- The data output we receive is a correlation between Terms and Documents
- NMF breaks down the multivariate data by creating a user-defined number of features. Each one of these features is a combination of the original attribute set. It is also key to remember these coefficients of these linear combinations are non-negative.

# Process Overview

Step 1: Select terms

Step 2: Select web documents

Step 3: Use web scraping to count the amount of terms in each document

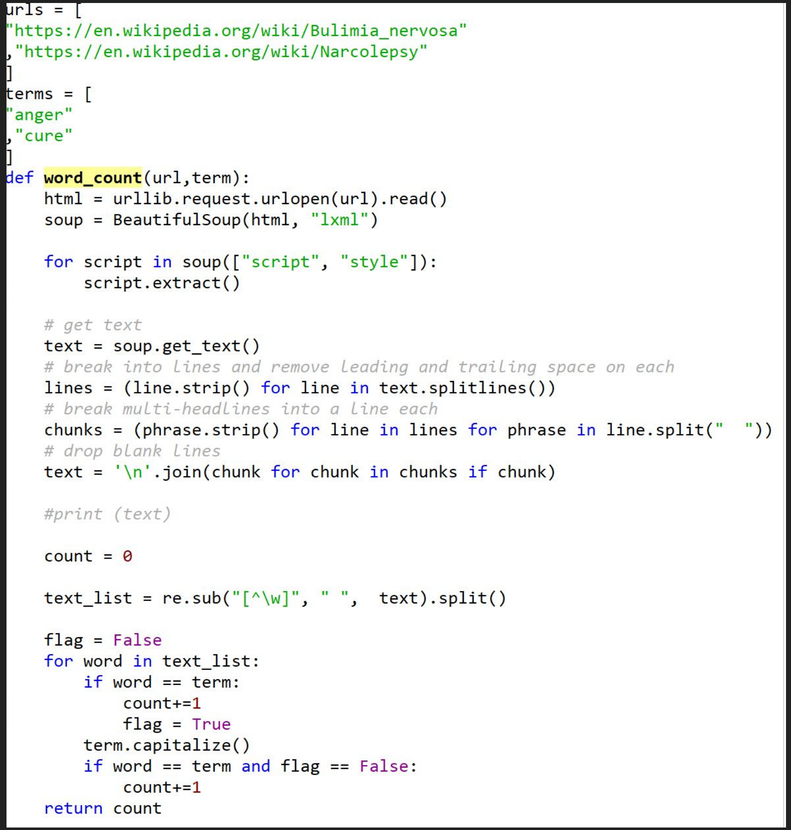
Step 4: Place web scraped matrix into Matlab NMF algorithm

Step 5: State the amount of iterations and clusters wanted

Step 6: Run NMF algorithm and evaluate data

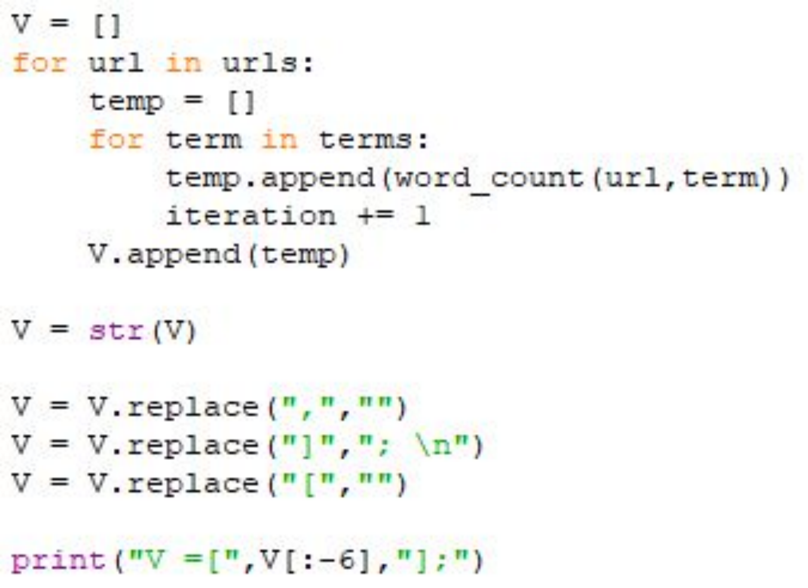
# Web Scrap Implementation

- Written in Python
- Word_count function used to automatically go to each website and count the number of times each term appeared
- The output was formatted to be placed directly into matlab

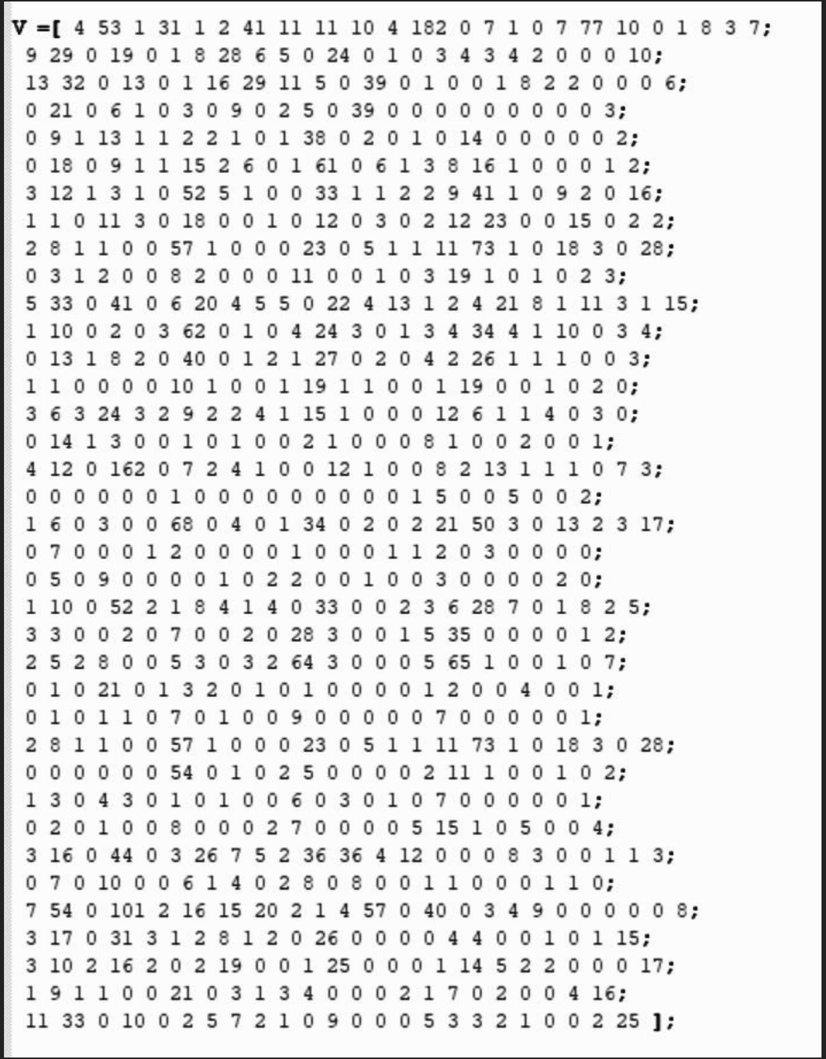


```
urls = [
"https://en.wikipedia.org/wiki/Bulimia_nervosa"
,"https://en.wikipedia.org/wiki/Narcolepsy"
]
terms = [
"anger"
,"cure"
]
def word_count(url,term):
    html = urllib.request.urlopen(url).read()
    soup = BeautifulSoup(html, "lxml")

    for script in soup(["script", "style"]):
        script.extract()

    # get text
    text = soup.get_text()
    # break into lines and remove leading and trailing space on each
    lines = (line.strip() for line in text.splitlines())
    # break multi-headlines into a line each
    chunks = (phrase.strip() for line in lines for phrase in line.split("  "))
    # drop blank lines
    text = '\n'.join(chunk for chunk in chunks if chunk)

    #print (text)

    count = 0

    text_list = re.sub("[^\w]", " ",  text).split()

    flag = False
    for word in text_list:
        if word == term:
            count+=1
            flag = True
        term.capitalize()
        if word == term and flag == False:
            count+=1
    return count
```

- Our original V matrix created from the web scrape code



```python
V = []
for url in urls:
    temp = []
    for term in terms:
        temp.append(word_count(url,term))
        iteration += 1
    V.append(temp)

V = str(V)

V = V.replace(",","")
V = V.replace("]","; \n")
V = V.replace("[","")

print("V =[",V[:-6],"];")
```

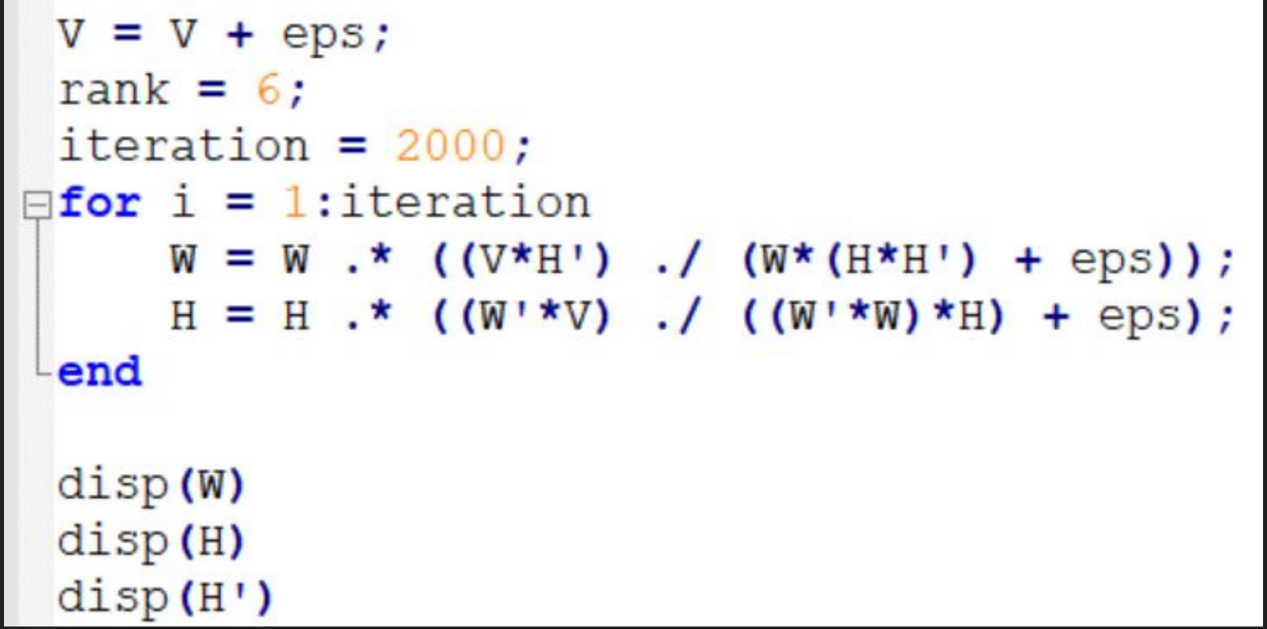


```
V =[ 4 53 1 31 1 2 41 11 11 10 4 182 0 7 1 0 7 77 10 0 1 8 3 7;
 9 29 0 19 0 1 8 28 6 5 0 24 0 1 0 3 4 3 4 2 0 0 0 10;
 13 32 0 13 0 1 16 29 11 5 0 39 0 1 0 0 1 8 2 2 0 0 0 6;
 0 21 0 6 1 0 3 0 9 0 2 5 0 39 0 0 0 0 0 0 0 0 0 3;
 0 9 1 13 1 1 2 2 1 0 1 38 0 2 0 1 0 14 0 0 0 0 0 2;
 0 18 0 9 1 1 15 2 6 0 1 61 0 6 1 3 8 16 1 0 0 0 1 2;
 3 12 1 3 1 0 52 5 1 0 0 33 1 1 2 2 9 41 1 0 9 2 0 16;
 1 1 0 11 3 0 18 0 0 1 0 12 0 3 0 2 12 23 0 0 15 0 2 2;
 2 8 1 1 0 0 57 1 0 0 0 23 0 5 1 1 11 73 1 0 18 3 0 28;
 0 3 1 2 0 0 8 2 0 0 0 11 0 0 1 0 3 19 1 0 1 0 2 3;
 5 33 0 41 0 6 20 4 5 5 0 22 4 13 1 2 4 21 8 1 11 3 1 15;
 1 10 0 2 0 3 62 0 1 0 4 24 3 0 1 3 4 34 4 1 10 0 3 4;
 0 13 1 8 2 0 40 0 1 2 1 27 0 2 0 4 2 26 1 1 1 0 0 3;
 1 1 0 0 0 0 10 1 0 0 1 19 1 1 0 0 1 19 0 0 1 0 2 0;
 3 6 3 24 3 2 9 2 2 4 1 15 1 0 0 0 12 6 1 1 4 0 3 0;
 0 14 1 3 0 0 1 0 1 0 0 2 1 0 0 0 8 1 0 0 2 0 0 1;
 4 12 0 162 0 7 2 4 1 0 0 12 1 0 0 8 2 13 1 1 1 0 7 3;
 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 5 0 0 5 0 0 2;
 1 6 0 3 0 0 68 0 4 0 1 34 0 2 0 2 21 50 3 0 13 2 3 17;
 0 7 0 0 0 1 2 0 0 0 0 1 0 0 0 1 1 2 0 3 0 0 0 0;
 0 5 0 9 0 0 0 0 1 0 2 2 0 0 1 0 0 3 0 0 0 0 2 0;
 1 10 0 52 2 1 8 4 1 4 0 33 0 0 2 3 6 28 7 0 1 8 2 5;
 3 3 0 0 2 0 7 0 0 2 0 28 3 0 0 1 5 35 0 0 0 0 1 2;
 2 5 2 8 0 0 5 3 0 3 2 64 3 0 0 0 5 65 1 0 0 1 0 7;
 0 1 0 21 0 1 3 2 0 1 0 1 0 0 0 0 1 2 0 0 4 0 0 1;
 0 1 0 1 1 0 7 0 1 0 0 9 0 0 0 0 0 7 0 0 0 0 0 1;
 2 8 1 1 0 0 57 1 0 0 0 23 0 5 1 1 11 73 1 0 18 3 0 28;
 0 0 0 0 0 0 54 0 1 0 2 5 0 0 0 0 2 11 1 0 0 1 0 2;
 1 3 0 4 3 0 1 0 1 0 0 6 0 3 0 1 0 7 0 0 0 0 0 1;
 0 2 0 1 0 0 8 0 0 0 2 7 0 0 0 0 5 15 1 0 5 0 0 4;
 3 16 0 44 0 3 26 7 5 2 36 36 4 12 0 0 0 8 3 0 0 1 1 3;
 0 7 0 10 0 0 6 1 4 0 2 8 0 8 0 0 1 1 0 0 0 1 1 0;
 7 54 0 101 2 16 15 20 2 1 4 57 0 40 0 3 4 9 0 0 0 0 0 8;
 3 17 0 31 3 1 2 8 1 2 0 26 0 0 0 0 4 4 0 0 1 0 1 15;
 3 10 2 16 2 0 2 19 0 0 1 25 0 0 0 1 14 5 2 2 0 0 0 17;
 1 9 1 1 0 0 21 0 3 1 3 4 0 0 0 2 1 7 0 2 0 0 4 16;
 11 33 0 10 0 2 5 7 2 1 0 9 0 0 0 5 3 3 2 1 0 0 2 25 ];
```

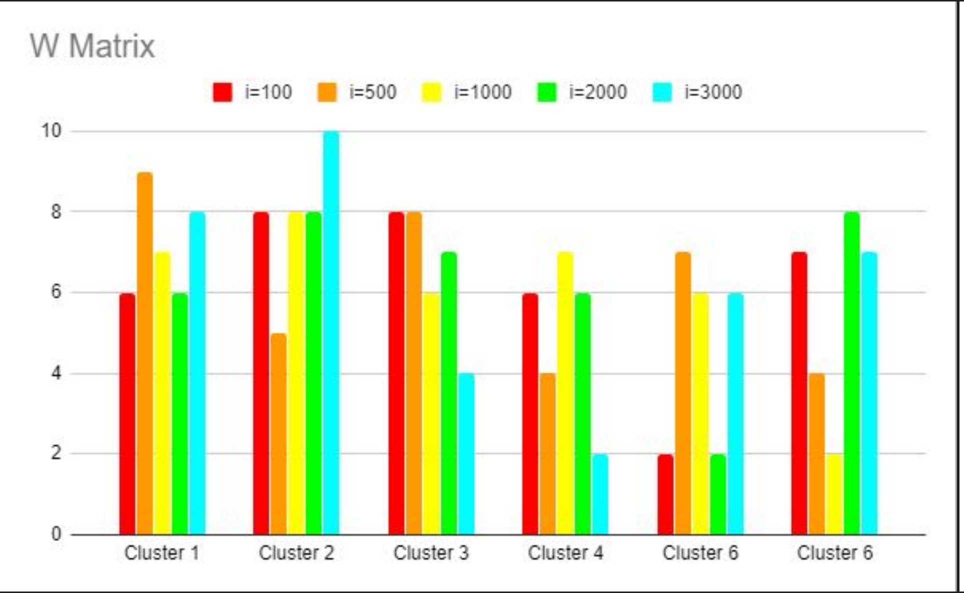
# NNMF Implementation

Matlab code

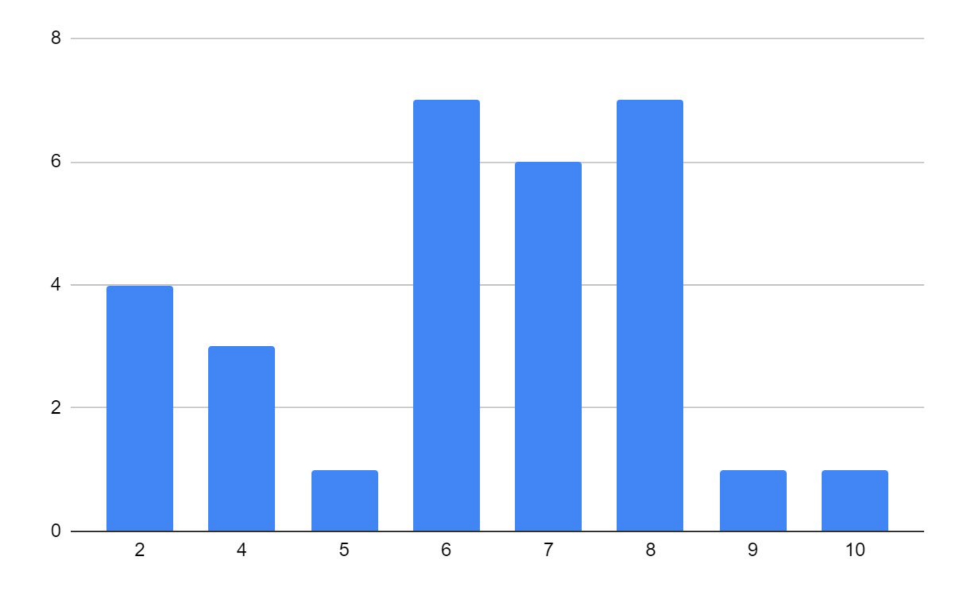
- Rank = clusters
- V = Original matix



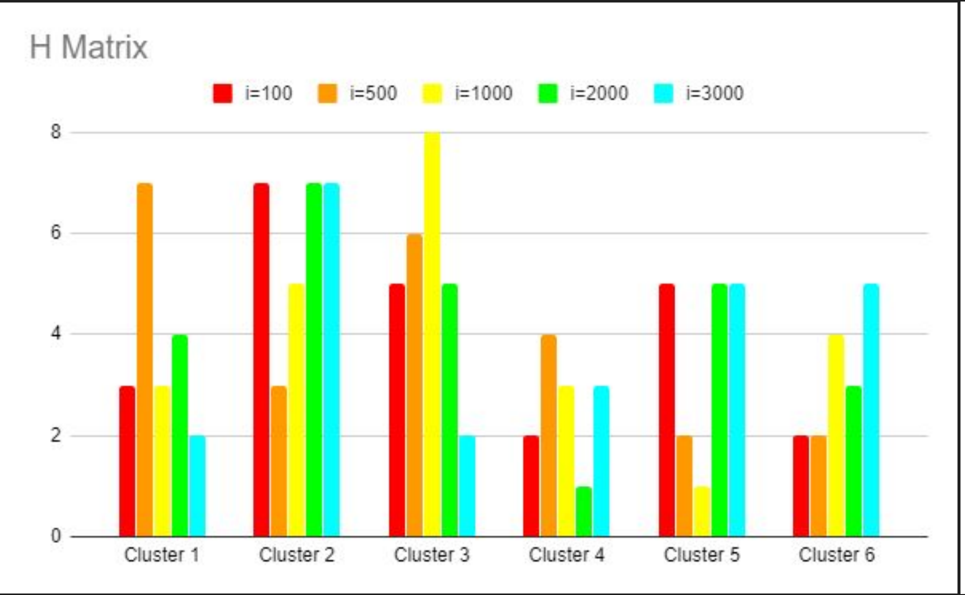
```matlab
V = V + eps;
rank = 6;
iteration = 2000;
for i = 1:iteration
    W = W .* ((V*H') ./ (W*(H*H') + eps));
    H = H .* ((W'*V) ./ ((W'*W)*H) + eps);
end

disp(W)
disp(H)
disp(H')
```



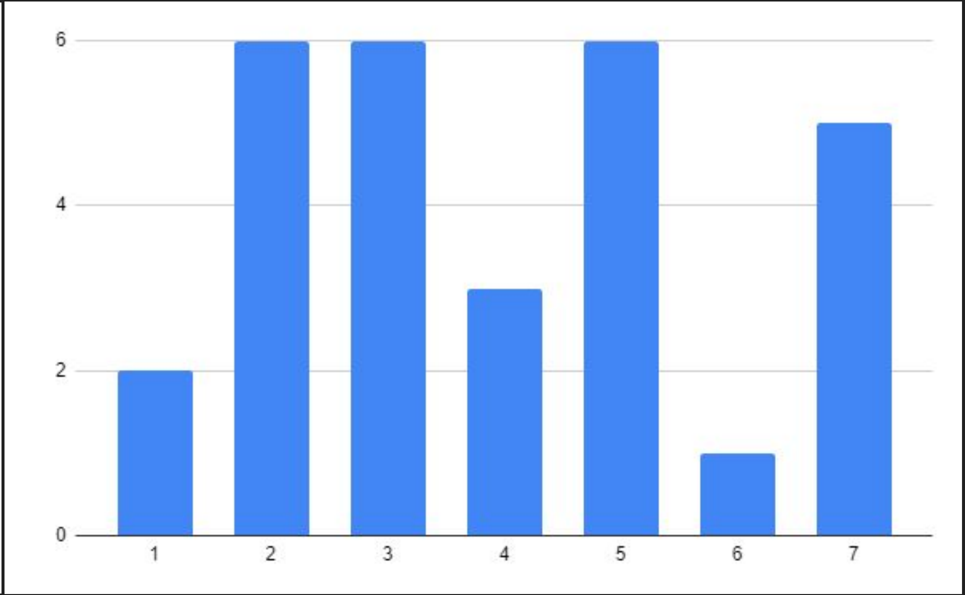
Size of each cluster through different iterations sizes



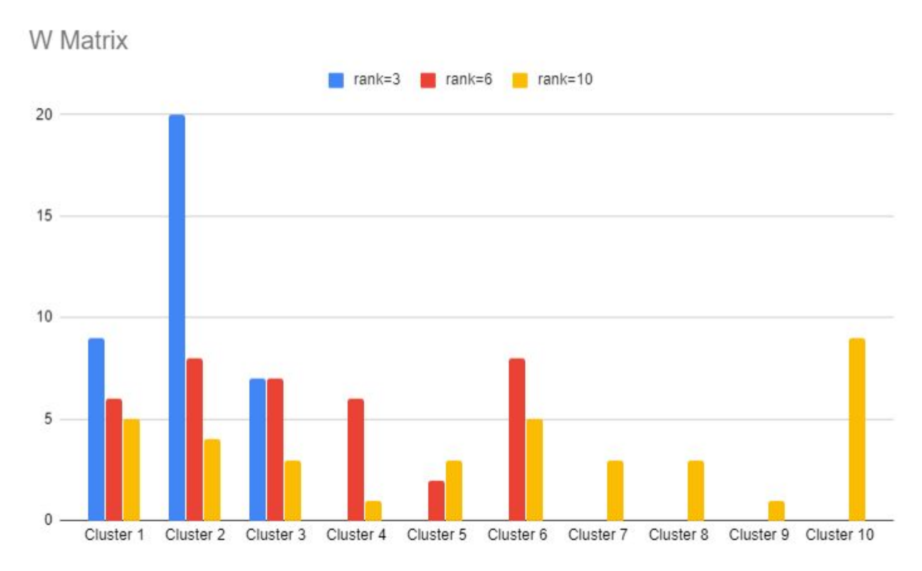
Number of times each cluster size appears



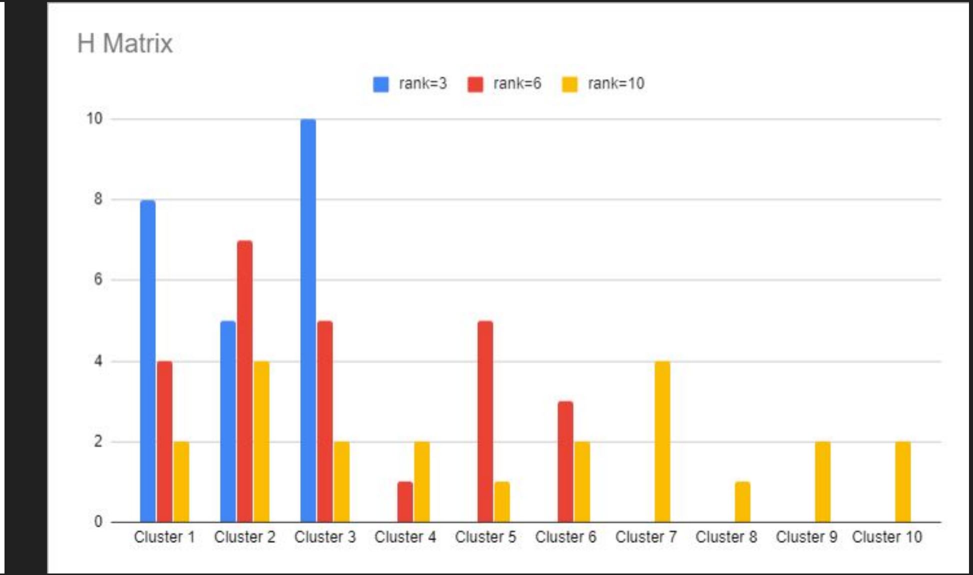
Size of each cluster through different iterations sizes



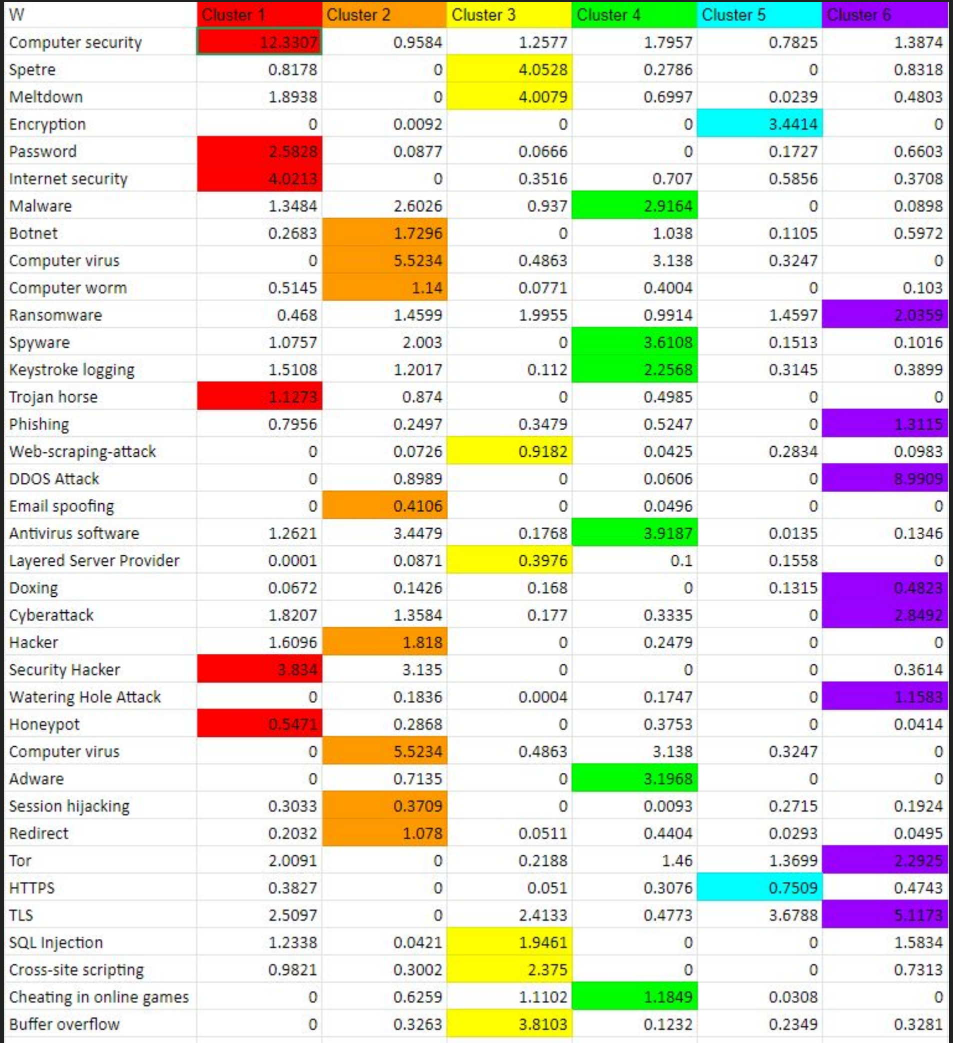
Number of times each cluster size appears



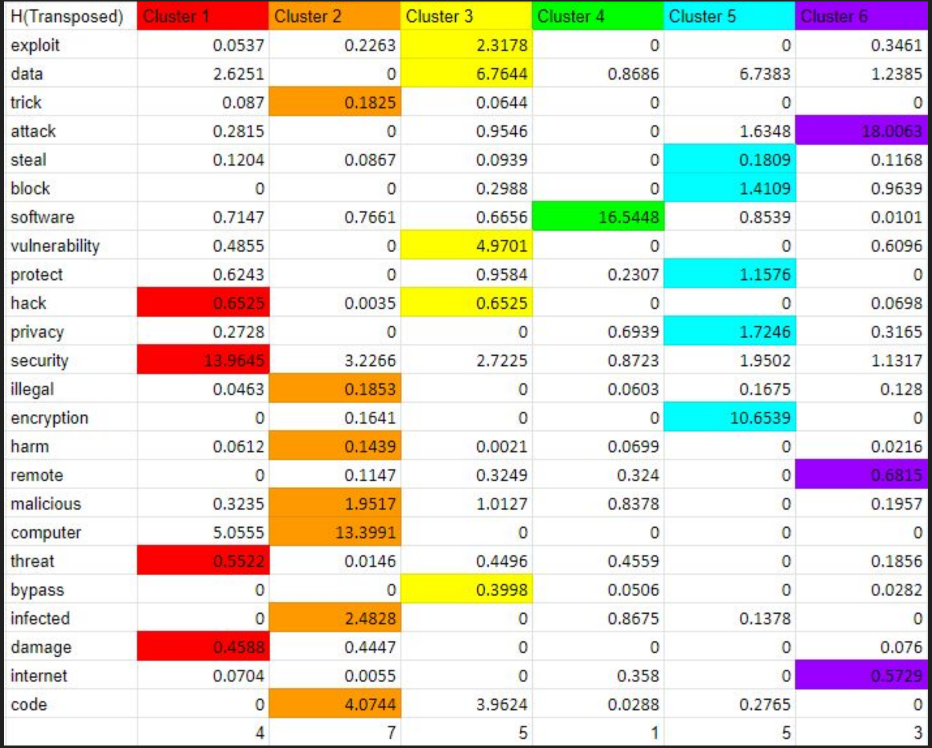
Size of each cluster through different rank sizes for W matrix



Size of each cluster through different rank sizes for H matrix

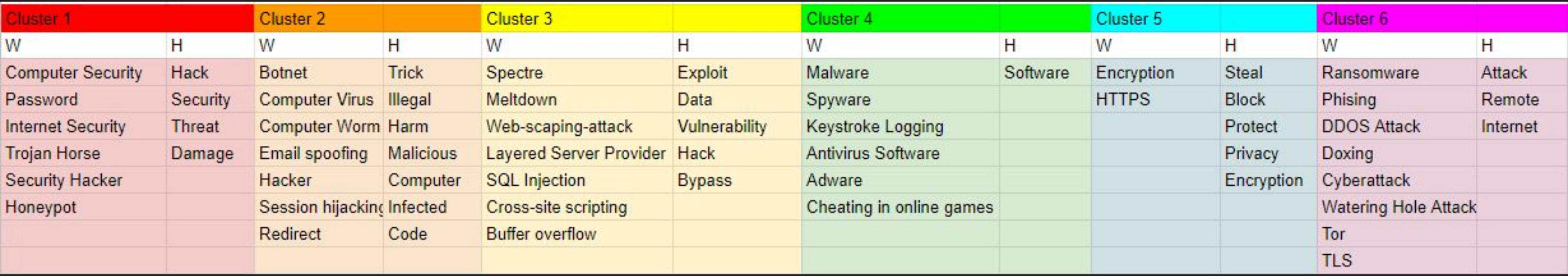


| W | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 |
|---|---|---|---|---|---|---|
| Computer security | 12.3307 | 0.9584 | 1.2577 | 1.7957 | 0.7825 | 1.3874 |
| Spetre | 0.8178 | 0 | 4.0528 | 0.2786 | 0 | 0.8318 |
| Meltdown | 1.8938 | 0 | 4.0079 | 0.6997 | 0.0239 | 0.4803 |
| Encryption | 0 | 0.0092 | 0 | 0 | 3.4414 | 0 |
| Password | 2.5828 | 0.0877 | 0.0666 | 0 | 0.1727 | 0.6603 |
| Internet security | 4.0213 | 0 | 0.3516 | 0.707 | 0.5856 | 0.3708 |
| Malware | 1.3484 | 2.6026 | 0.937 | 2.9164 | 0 | 0.0898 |
| Botnet | 0.2683 | 1.7296 | 0 | 1.038 | 0.1105 | 0.5972 |
| Computer virus | 0 | 5.5234 | 0.4863 | 3.138 | 0.3247 | 0 |
| Computer worm | 0.5145 | 1.14 | 0.0771 | 0.4004 | 0 | 0.103 |
| Ransomware | 0.468 | 1.4599 | 1.9955 | 0.9914 | 1.4597 | 2.0359 |
| Spyware | 1.0757 | 2.003 | 0 | 3.6108 | 0.1513 | 0.1016 |
| Keystroke logging | 1.5108 | 1.2017 | 0.112 | 2.2568 | 0.3145 | 0.3899 |
| Trojan horse | 1.1273 | 0.874 | 0 | 0.4985 | 0 | 0 |
| Phishing | 0.7956 | 0.2497 | 0.3479 | 0.5247 | 0 | 1.3115 |
| Web-scraping-attack | 0 | 0.0726 | 0.9182 | 0.0425 | 0.2834 | 0.0983 |
| DDOS Attack | 0 | 0.8989 | 0 | 0.0606 | 0 | 8.9909 |
| Email spoofing | 0 | 0.4106 | 0 | 0.0496 | 0 | 0 |
| Antivirus software | 1.2621 | 3.4479 | 0.1768 | 3.9187 | 0.0135 | 0.1346 |
| Layered Server Provider | 0.0001 | 0.0871 | 0.3976 | 0.1 | 0.1558 | 0 |
| Doxing | 0.0672 | 0.1426 | 0.168 | 0 | 0.1315 | 0.4823 |
| Cyberattack | 1.8207 | 1.3584 | 0.177 | 0.3335 | 0 | 2.8492 |
| Hacker | 1.6096 | 1.818 | 0 | 0.2479 | 0 | 0 |
| Security Hacker | 3.834 | 3.135 | 0 | 0 | 0 | 0.3614 |
| Watering Hole Attack | 0 | 0.1836 | 0.0004 | 0.1747 | 0 | 1.1583 |
| Honeypot | 0.5471 | 0.2868 | 0 | 0.3753 | 0 | 0.0414 |
| Computer virus | 0 | 5.5234 | 0.4863 | 3.138 | 0.3247 | 0 |
| Adware | 0 | 0.7135 | 0 | 3.1968 | 0 | 0 |
| Session hijacking | 0.3033 | 0.3709 | 0 | 0.0093 | 0.2715 | 0.1924 |
| Redirect | 0.2032 | 1.078 | 0.0511 | 0.4404 | 0.0293 | 0.0495 |
| Tor | 2.0091 | 0 | 0.2188 | 1.46 | 1.3699 | 2.2925 |
| HTTPS | 0.3827 | 0 | 0.051 | 0.3076 | 0.7509 | 0.4743 |
| TLS | 2.5097 | 0 | 2.4133 | 0.4773 | 3.6788 | 5.1173 |
| SQL Injection | 1.2338 | 0.0421 | 1.9461 | 0 | 0 | 1.5834 |
| Cross-site scripting | 0.9821 | 0.3002 | 2.375 | 0 | 0 | 0.7313 |
| Cheating in online games | 0 | 0.6259 | 1.1102 | 1.1849 | 0.0308 | 0 |
| Buffer overflow | 0 | 0.3263 | 3.8103 | 0.1232 | 0.2349 | 0.3281 |



| H(Transposed) | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 |
|---|---|---|---|---|---|---|
| exploit | 0.0537 | 0.2263 | 2.3178 | 0 | 0 | 0.3461 |
| data | 2.6251 | 0 | 6.7644 | 0.8686 | 6.7383 | 1.2385 |
| trick | 0.087 | 0.1825 | 0.0644 | 0 | 0 | 0 |
| attack | 0.2815 | 0 | 0.9546 | 0 | 1.6348 | 18.0063 |
| steal | 0.1204 | 0.0867 | 0.0939 | 0 | 0.1809 | 0.1168 |
| block | 0 | 0 | 0.2988 | 0 | 1.4109 | 0.9639 |
| software | 0.7147 | 0.7661 | 0.6656 | 16.5448 | 0.8539 | 0.0101 |
| vulnerability | 0.4855 | 0 | 4.9701 | 0 | 0 | 0.6096 |
| protect | 0.6243 | 0 | 0.9584 | 0.2307 | 1.1576 | 0 |
| hack | 0.6525 | 0.0035 | 0.6525 | 0 | 0 | 0.0698 |
| privacy | 0.2728 | 0 | 0 | 0.6939 | 1.7246 | 0.3165 |
| security | 13.9645 | 3.2266 | 2.7225 | 0.8723 | 1.9502 | 1.1317 |
| illegal | 0.0463 | 0.1853 | 0 | 0.0603 | 0.1675 | 0.128 |
| encryption | 0 | 0.1641 | 0 | 0 | 10.6539 | 0 |
| harm | 0.0612 | 0.1439 | 0.0021 | 0.0699 | 0 | 0.0216 |
| remote | 0 | 0.1147 | 0.3249 | 0.324 | 0 | 0.6815 |
| malicious | 0.3235 | 1.9517 | 1.0127 | 0.8378 | 0 | 0.1957 |
| computer | 5.0555 | 13.3991 | 0 | 0 | 0 | 0 |
| threat | 0.5522 | 0.0146 | 0.4496 | 0.4559 | 0 | 0.1856 |
| bypass | 0 | 0 | 0.3998 | 0.0506 | 0 | 0.0282 |
| infected | 0 | 2.4828 | 0 | 0.8675 | 0.1378 | 0 |
| damage | 0.4588 | 0.4447 | 0 | 0 | 0 | 0.076 |
| internet | 0.0704 | 0.0055 | 0 | 0.358 | 0 | 0.5729 |
| code | 0 | 4.0744 | 3.9624 | 0.0288 | 0.2765 | 0 |
|  | 4 | 7 | 5 | 1 | 5 | 3 |

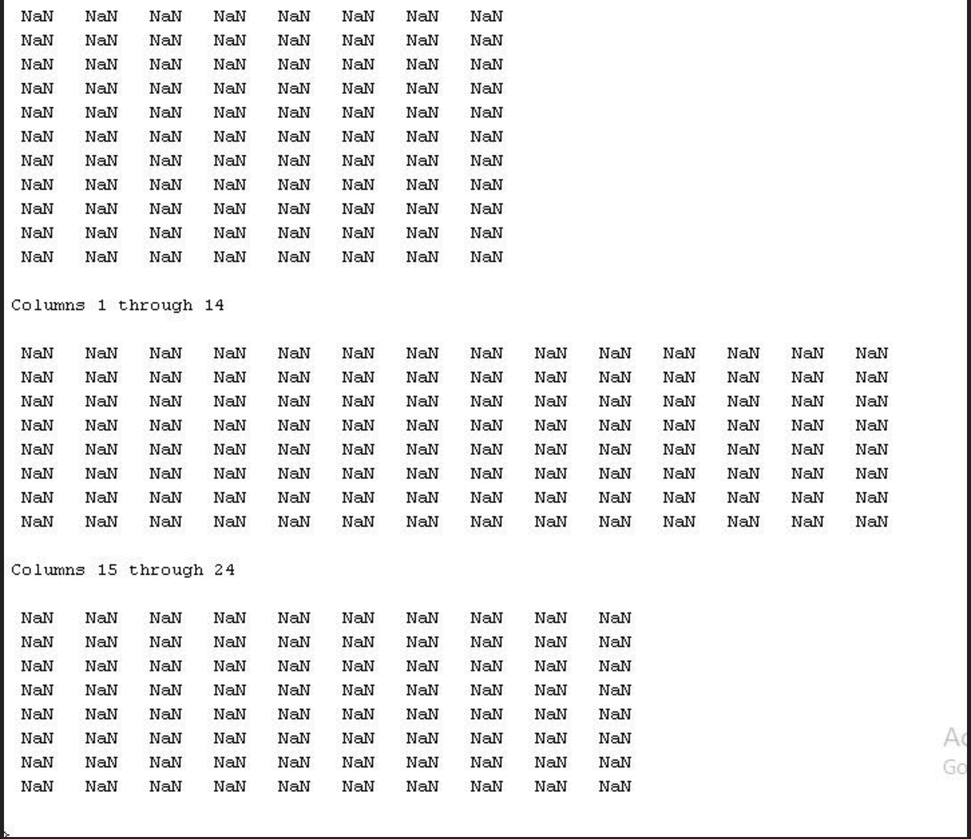
# Final Output



| Cluster 1 | | Cluster 2 | | Cluster 3 | | Cluster 4 | | Cluster 5 | | Cluster 6 | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| W | H | W | H | W | H | W | H | W | H | W | H |
| Computer Security | Hack | Botnet | Trick | Spectre | Exploit | Malware | Software | Encryption | Steal | Ransomware | Attack |
| Password | Security | Computer Virus | Illegal | Meltdown | Data | Spyware | | HTTPS | Block | Phising | Remote |
| Internet Security | Threat | Computer Worm | Harm | Web-scaping-attack | Vulnerability | Keystroke Logging | | | Protect | DDOS Attack | Internet |
| Trojan Horse | Damage | Email spoofing | Malicious | Layered Server Provider | Hack | Antivirus Software | | | Privacy | Doxing | |
| Security Hacker | | Hacker | Computer | SQL Injection | Bypass | Adware | | | Encryption | Cyberattack | |
| Honeypot | | Session hijacking | Infected | Cross-site scripting | | Cheating in online games | | | | Watering Hole Attack | |
| | | Redirect | Code | Buffer overflow | | | | | | Tor | |
| | | | | | | | | | | TLS | |

# Pitfalls

- We had many 0’s in our original matrix making for NaN output for our W and H matrices
  - Solution: we needed to add eps (epsilon) to our V matrix



```
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

Columns 1 through 14

   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN

Columns 15 through 24

   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN   NaN
```

# Works cited

https://iksinc.online/2016/03/21/what-is-nmf-and-what-can-you-do-with-it/

https://www.youtube.com/watch?reload=9&v=ZTxXGZwe2gw

https://www.youtube.com/watch?v=I3cjbB38Z4A

https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_nmf.htm#CHDEHCGC

https://docs.oracle.com/cd/B28359_01/datamine.111/b28129/text.htm#CIHGGBEI

https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

https://en.wikipedia.org/wiki/Document-term_matrix

https://en.wikipedia.org/wiki/Tf–idf

https://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-pyth on/

https://epubs.siam.org/doi/pdf/10.1137/1.9781611972740.45