Hanna Foerster
Lzhc88

Data collection and cleaning assignment report

In the following I will explain the program that I have created to calculate the distance between the 10 keywords: ‘targeted threat’, ‘Advanced Persistent Threat’, ‘phishing’, ‘DoS attack’, ‘malware’, ‘computer virus’, ‘spyware’, ‘malicious bot’, ‘ransomware’ and ‘encryption’.

First, in problem 1 and 2 the BBC website was searched for each keyword and the content of the top 100 articles for each keyword were copied into a separate text file each. For web scraping, to get the URLs and the text content of the articles, the BeautifulSoup package was utilized. To get the relevant articles for each keyword, the search bar of the BBC website was used and then from all BBC content that was included in the search result only the articles were filtered out. However, there were still some irrelevant articles. To combat this, initially I only took articles that included the keyword in the title. This did not give enough articles however and filtered out many relevant results, so I dropped this idea. Then I tried searching the whole text body for the keyword, however this took a lot of computational time and still filtered out many relevant results. Therefore, in the end only the BBC search function was used to get the relevant results. All articles that were turned into text files were saved into a folder ‘articles’ in the same directory as the script and were named with the keyword and the search result ranking. This part of the program takes quite a lot of computational time, because hundreds of articles’ contents must be copied and saved into text files.

In problem 3 the semantic distance between the keywords was calculated. Many methods of measuring distances have been suggested. For example, Princeton’s wordnet [1] measures the semantic closeness of words by meaning but also by relationship such as hyponymy relationships. So, words similar in meaning that would also be given by a thesaurus and words that have subordinate relationships such as ‘tree’ and ‘leaves’ would be seen as close. Then to measure the distance between dissimilar words a graph distance approach using wordnet could be applied. Another concept would be to define two words to be semantically close if they often appear in the same context. This concept was introduced by Firth [2] and is called distributional hypothesis. Following this hypothesis, a technique called Latent Semantic Analysis (LSA) was developed. It is described as a statistical technique that takes as input raw text and represents this text as a matrix of all words as columns and contexts as indices. Each cell then contains the frequency at which these words appear in each context [3].

I have followed a similar approach for my algorithm. The first method with which I calculated the semantic distance is with the number of occurrences of the keywords in the BBC articles of one keyword. To save the count of occurrences a matrix was used with column, article of keyword i and row, keyword j. If keyword j was mentioned in keyword i’s article then 1 was added to the matrix value at [i,j]. After running this algorithm some cells of the occurrence matrix still had the value 0 (Fig. 1), denoting no occurrences of this keyword found in the articles of keyword i. If I were to calculate the distance between words with cell value 0, I would have to label them maximally distant, but between word pairs with maximum distance, I would not know which word pairs were more dissimilar. Hence, I decided to add to the occurrence matrix the number of mentions of the keywords in the Wikipedia article of each keyword. So, if for example ‘malicious bot’ was mentioned three times in the Wikipedia

[1]: Princeton University. "About WordNet." WordNet. Princeton University. 2010.
[2]: Firth, J.R. "A synopsis of linguistic theory 1930-1955." Studies in Linguistic Analysis, Special volume of the Philological Society, Oxford, The Philological Society, 1957, pp. 1-32.
[3]: Landauer, Thomas K., et al. "An Introduction to Latent Semantic Analysis.” Discourse Processes, vol. 25, no. 2-3, 1998, pp. 259-84. Crossref, doi: 10.1080/01638539809545028

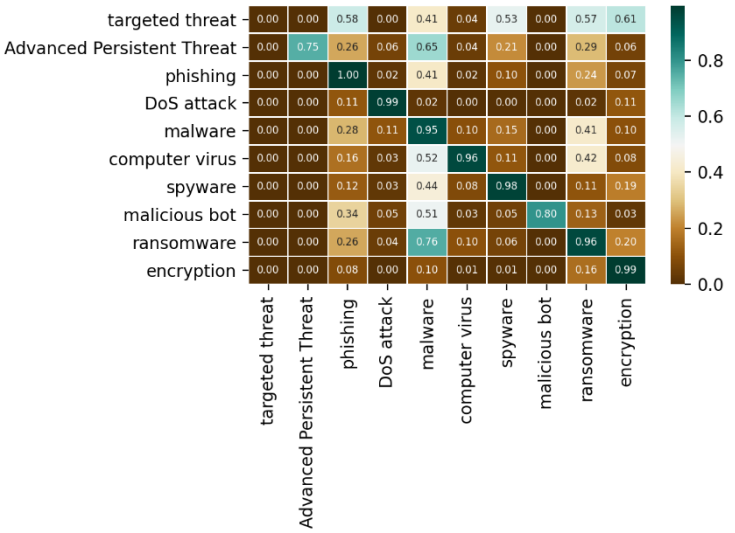

Figure 1: Heatmap with article keyword frequencies: Method 1.1

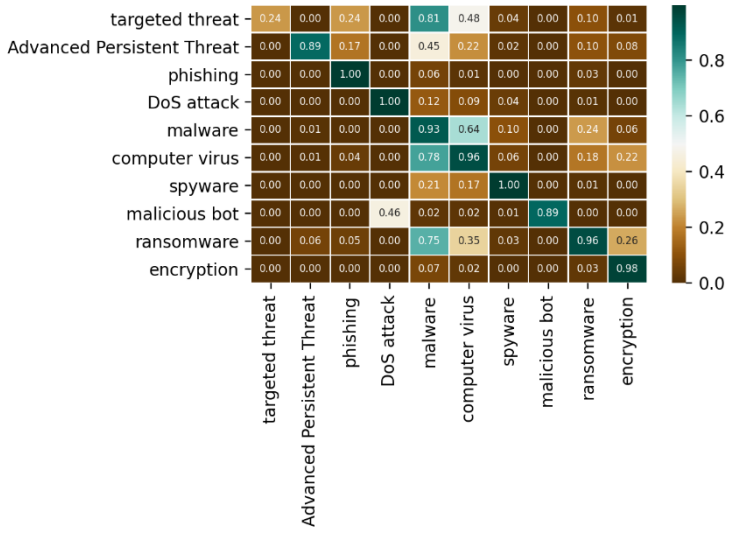

Figure 2: Heatmap with Wikipedia keyword frequencies: Method 1.2

article of 'targeted threat', then 3 would be added to the corresponding cell in the occurrence matrix (Fig. 2). Now, since word similarities should be the same both ways, i.e. similarity from 'malicious bot' to 'targeted threat' should be the same as similarity from 'targeted threat' to malicious bot', cell values [i,j] were added to [j,i] and conversely cell values [j,i] were added to [i,j]. Then, I used cosine similarity as a measure to calculate similarity. Cosine similarity is calculated with the following formula:

$$similarity = \cos(\Theta) = \frac{\Sigma_1^n A_i B_i}{\sqrt{\Sigma_1^n A^2{}_i} * \sqrt{\Sigma_1^n B^2{}_i}}$$

The cosine similarity was calculated between each row and each column, so that each cell in the matrix had an individual similarity value. I denote this first method of calculating semantic similarity 'Method 1' (Fig. 3), it combines the two approaches I have just described.

Since, some words still had no similarities between each other, I developed two more methods, which are very similar to LSA. The text content of each keyword's article was split into words and a word count in dictionary form was taken for each of the words. This was repeated for 9 articles for each keyword. It was only done for 9 articles each because the keyword 'malicious bot' only had 9 articles and if all 100 articles would have been taken for some keywords, their word count would have been much bigger and thus not comparable with the word count of 'malicious bot'. So, for each keyword I ended up with a big dictionary with word counts. I combined these dictionaries into a data frame, which had a column for all words from the dictionaries and a row for each keyword. Then the cosine similarity was calculated between each row, i.e., each keyword's word counts. This gave me a symmetric matrix with similarities between each word. Since many fill words in the articles were the same the similarity between keywords was higher than expected and all similarity values lay in a small range. Hence, min-max scaling was applied (Fig.4). As a last method the Wikipedia article of the keywords were compared to each other in a similar way. The number of words that were analysed for each article was equal to the length of the shortest Wikipedia article. Similarity was again calculated by cosine similarity and min-max normalization was applied (Fig. 5).

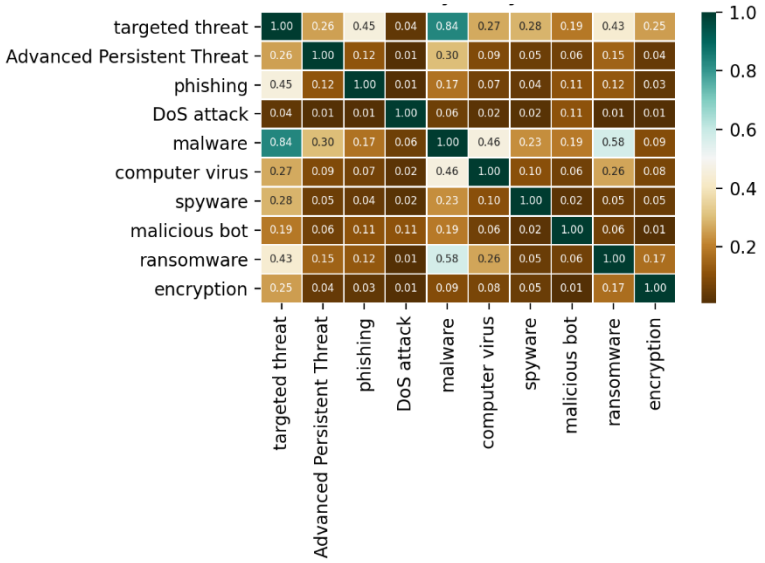

Figure 3: Heatmap keyword similarities: Method 1

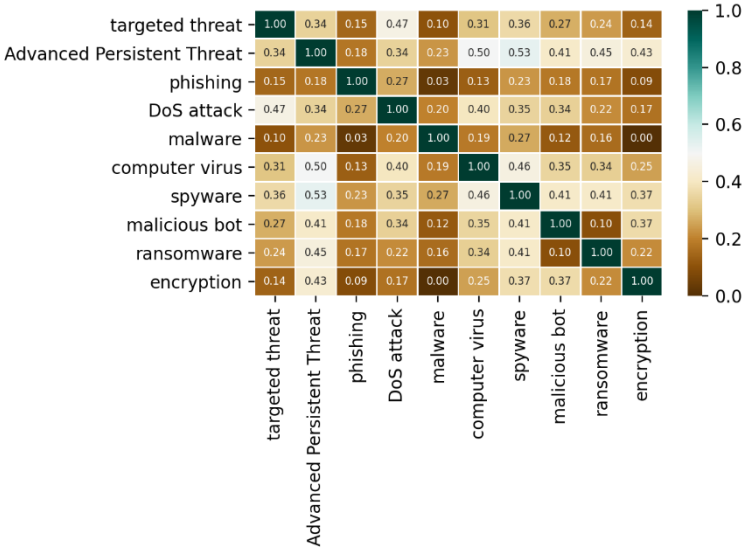

Figure 4: Heatmap LSA of 9 articles: Method 2

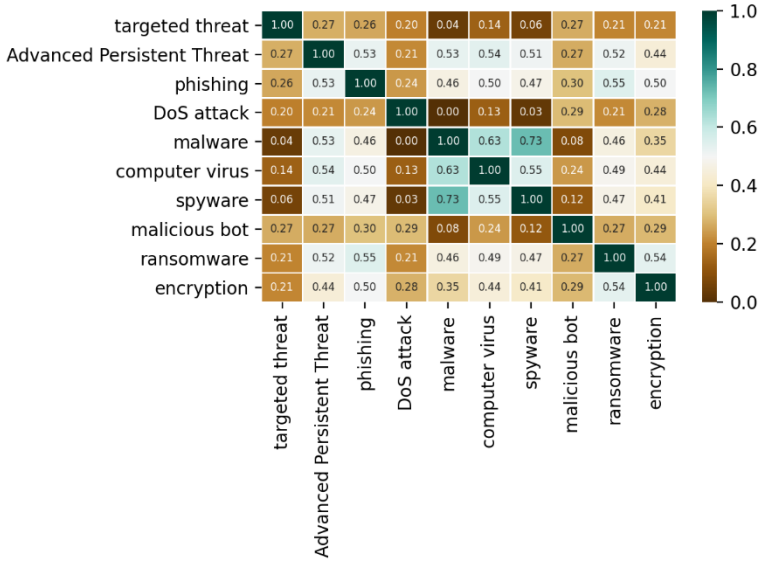

Figure 5: Heatmap LSA of Wikipedia articles: Method 3

Lastly, I combined the three similarity matrices that I had calculated with the weights 0.5 for the first method and 0.25 for the last two methods (Fig.6). Combining all methods lead to a less biased measure of distance. If I had for example only taken the keyword occurrence frequencies into account (i.e., Method 1) then the distance measure would not have captured all of the contextual relationship between the keywords. Moreover, including the Wikipedia article as an object of analysis ensured that the keywords deemed as close were also close in meaning and not only often mentioned in a similar context. Overall, because four different methods were applied to calculate the similarity, the computational time was four times longer than if only one method had been applied. However, outliers in individual results of the methods could be cross-checked and a less biased measure of similarity was found.

To get the distance instead of the similarity, 1 minus the similarity value was taken. As a final step the distance data frame was saved as an excel sheet called ‘distance.xlsx’. The distances were between 0 and 1. The keywords with the largest distances were ‘malware’ and ‘DoS attack’, ‘spyware and ‘DoS attack’, ‘ransomware’ and ‘malicious bot’, ‘encryption’ and ‘DoS attack’, and ‘malicious bot’ and ‘ransomware’. Generally, ‘DoS attack’ and ‘malicious bot’ have the least similarity to the other keywords. The words that were closest to each other were ‘malware’ and ‘targeted threat’, ‘malware’ and ‘ransomware’, and ‘malware’ and ‘computer virus’ (Fig. 7).

In problem 4 we were supposed to visualise the distances we got. I have chosen to do this with a heatmap, because it shows the distances in a way that they can be easily compared to each other.

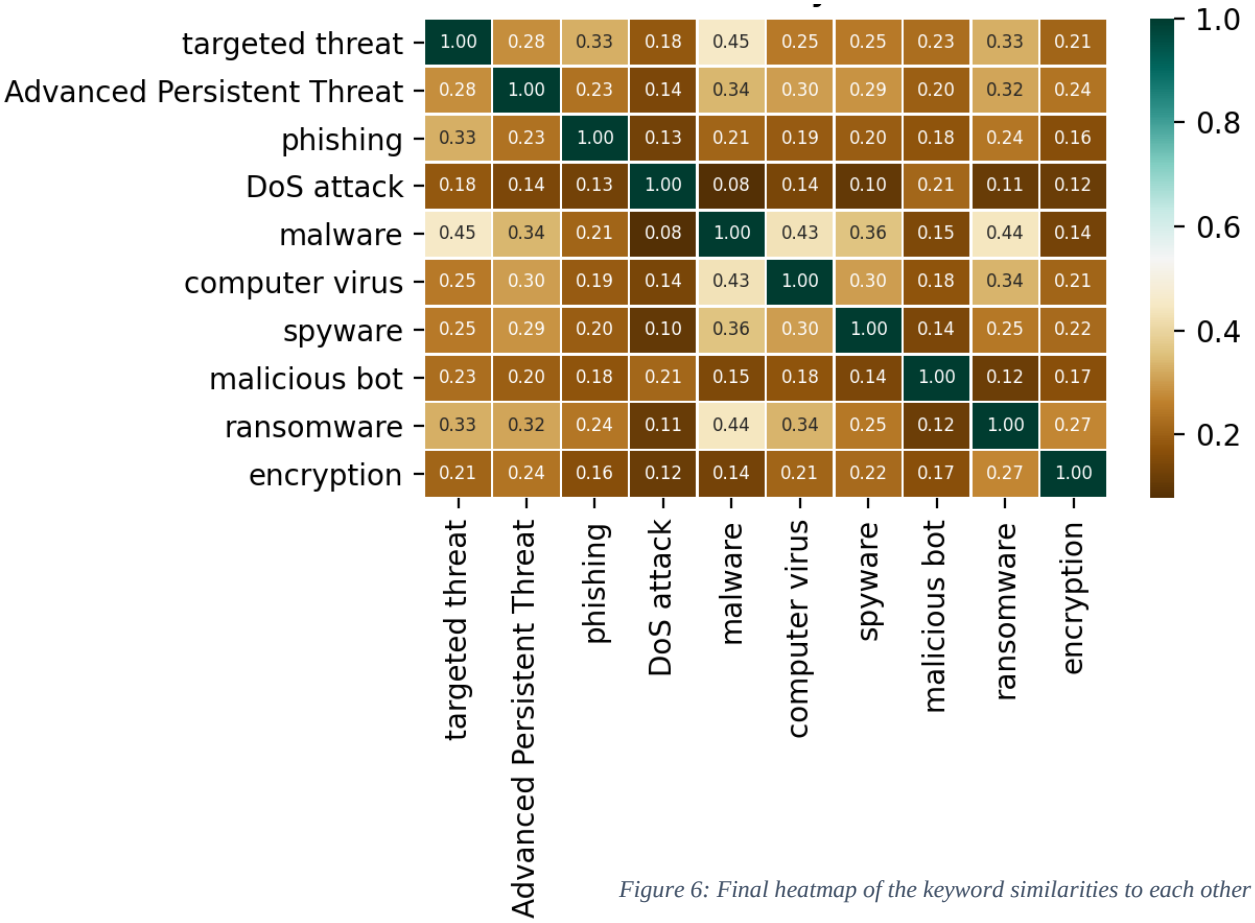

Figure 6: Final heatmap of the keyword similarities to each other

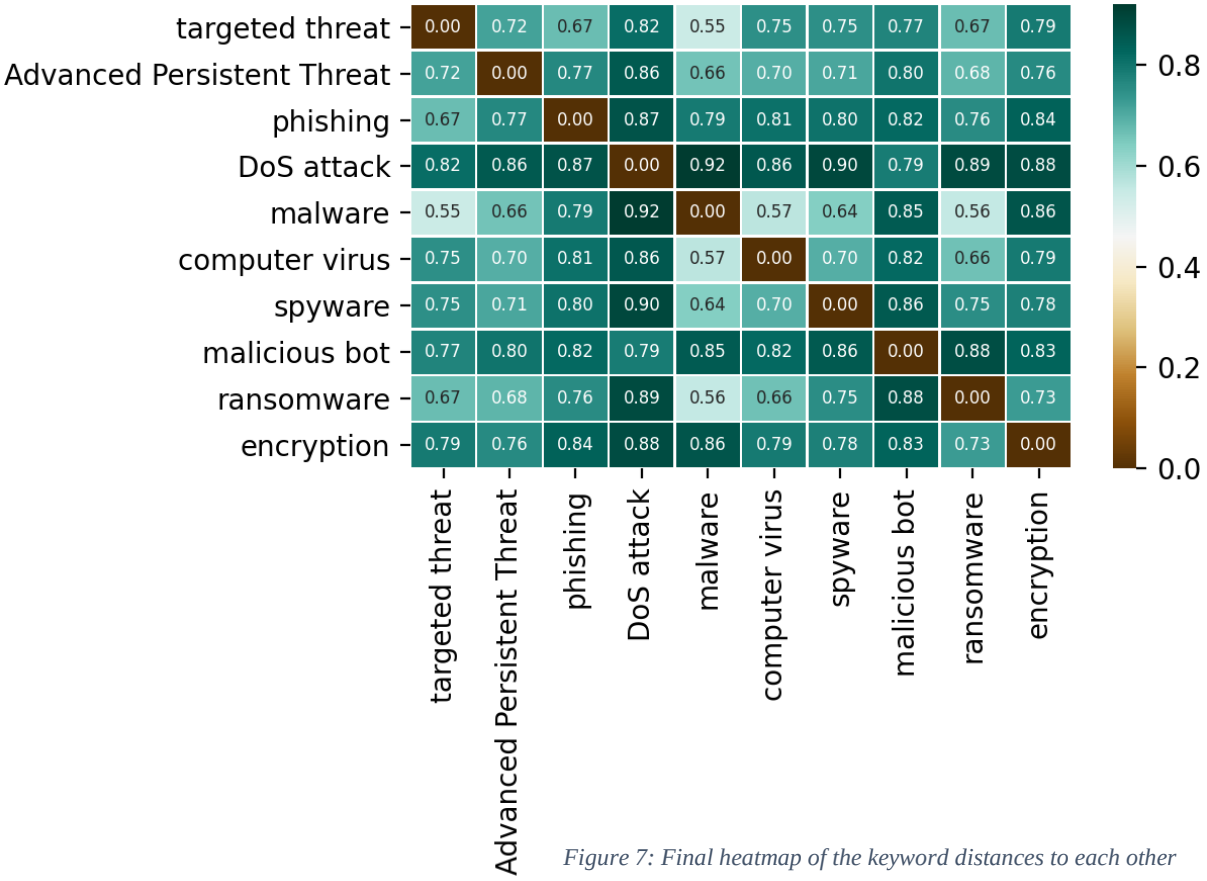

Figure 7: Final heatmap of the keyword distances to each other