Semilarity: Calculating Semantic Distances between news articles

1 Introduction

The program `main.py` uses data collection and cleaning techniques to download articles from BBC News and then calculate and visualise their semantic distances based on keyword used to search.

All required libraries are in `requirements.txt` and in the comments of the python program. If needed, download all libraries using `pip install -r requirements.txt`. The program `main.py` **must be in the same directory** as `keywords.xlsx`. On average, the entire program takes approximately **13 minutes**[1] to run.

2 Methods

The articles were found, accessed and downloaded as `.txt` files through the search bar on `www.bbc.co.uk/news` with the help of the `requests` and `urllib` standard python libraries for Problem 1. All 10 keywords were read in from cells A2 to A11 of `keywords.xlsx`. The Beautiful Soup library (`bs4`) was used to collect and process the text from the articles in these webpages and stored them in the algorithm's corpus data structure, using a newly defined `html_to_text()` function for Problem 2. The first 100 articles were collected in the order of BBC's search relevance rankings, and the algorithm had a filtering mechanism to only download BBC News articles, and not irrelevant search results (BBC Sports, Programmes, Bitesize etc). For keywords with fewer than 100 BBC News article results, the maximum possible number of BBC News articles were scraped, and any potential bias is handled by the error checking measures outlined at the end of this section.

Keyword	Article Count	Keyword	Article Count
targeted threat	100	computer virus	100
Advanced Persistent Threat	12	spyware	100
phishing	100	malicious bot	9
DoS attack	100	ransomware	100
malware	100	encryption	100

Table 1: Number of articles scraped for each keyword searched on `www.bbc.co.uk/news`

To calculate and quantify the *semantic distances* between keywords using articles in the corpus for Problem 3, the articles had to be represented as mathematical objects to better abstract the problem. First, articles collected for a given keyword were collated into a document for that keyword. The collection of 10 documents together formed the corpus, which was then loaded into the `documents` column of the `articles_df` dataframe. Dataframes were used for ease of data manipulation before any processing or embedding. Next, the documents themselves were cleaned of characters and stop-words that did not contribute to the semantic distance, similarity or meaning of the articles. By removing stop-words that were repeated across articles (like 'Published', 'and', 'is' etc), the algorithm focused on semantically meaningful relations between articles [2]. This was achieved by the algorithm's use of the standard python `re` library for regular expressions (regex), the results of which were stored in the `documents_cleaned` column of `articles_df`.

The next stage of the algorithm for problem 3 involved the creation of an *unsupervised neural network* to facilitate the representation and analysis of the keyword documents as document vectors. A model for storing documents as 100-dimensional vectors was created by leveraging Doc2Vec functionalities of the open-source `gensim` python library. Doc2Vec facilitates a tried and tested framework for semantic distance calculations [1], and was chosen over other machine learning algorithms (such as Word2Vec) due to it's semantic accuracy, low algorithmic error rate and efficiency in handling large volumes of documents. Word2Vec would have crudely approximated the documents vectors for each keyword by averaging word vectors present in the articles.

The model was then trained using the publicly available `punkt` tokenizer from the `nltk` (Natural Language ToolKit) python library. This allowed the model to learn the basic semantics of the English language before studying the BBC articles. The program then tokenized the articles from the cleaned documents and used it to train the model by embedding the documents as document vectors in a 100 dimensional document vector space. The algorithm stored these vectors in `document_embeddings` data structure, which was accessible to the `sklearn` python library for linear algebra calculations.

Finally, the algorithm was able to calculate the Euclidean distances (the $p = 2$ case of Minkowski distances) between the 10 document vectors in this semantic vector space to produce an adjacency matrix of the relative *semantic distances*

[1] tested on an Intel Core i5-1035G4 1.1GHz quad core CPU

between keywords,

$$\text{semantic distance}(\vec{\mathbf{u}}, \vec{\mathbf{v}}) = \sqrt{\sum_{i=1}^{100} (u_i - v_i)^2},$$

with u_i and v_i being the i^{th} semantic component of document vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$. A semantic distance of 0 indicates identical articles and larger semantic distances indicate articles are less semantically related. The program exports the adjacency matrix and saves it to the newly created `distances.xlsx` file.

Additionally, the *cosine similarity* along with the *Term Frequency-Inverse Document Frequency* (TF-IDF) between document vectors were also calculated for error and sanity checking the results of the data visualization.

$$\text{cosine similarity}(\vec{\mathbf{u}}, \vec{\mathbf{v}}) = \frac{\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}}{|\vec{\mathbf{u}}||\vec{\mathbf{v}}|} = \cos\theta,$$

where cosine similarity$(\vec{\mathbf{u}}, , \vec{\mathbf{v}})$ is the similarity score between two normalised document vectors $\vec{\mathbf{u}}$ and $\vec{\mathbf{v}}$ which together enclose and acute angle θ. Cosine similarity can range from 0 (orthogonal vectors with absolutely no semantic similarity) to 1 (exactly identical documents), and should be interpretted in conjunction with the semantic distances in `distances.xlsx` from Problem 3, as shown in Fig 2.

3 Results and Discussions (Visualisations for Problem 4)

The following figures visualise the results of the algorithm. The *semantic distances* between keywords have been visualised in Fig.1. After multiple runs of the program, most articles in the corpus are within 30 units of eachother. The distribution and spread of distance and cosine similarity scores are shown in Fig. 2. Cosine similarity scores show a more narrow range of values than semantic distances, which is to be expected since all of the keywords used were related to cybersecurity. Slight variations in semantic distances or similarity scores between runs are caused by variations in the seed for training and testing splitting by the non-deterministic unsupervised neural network.

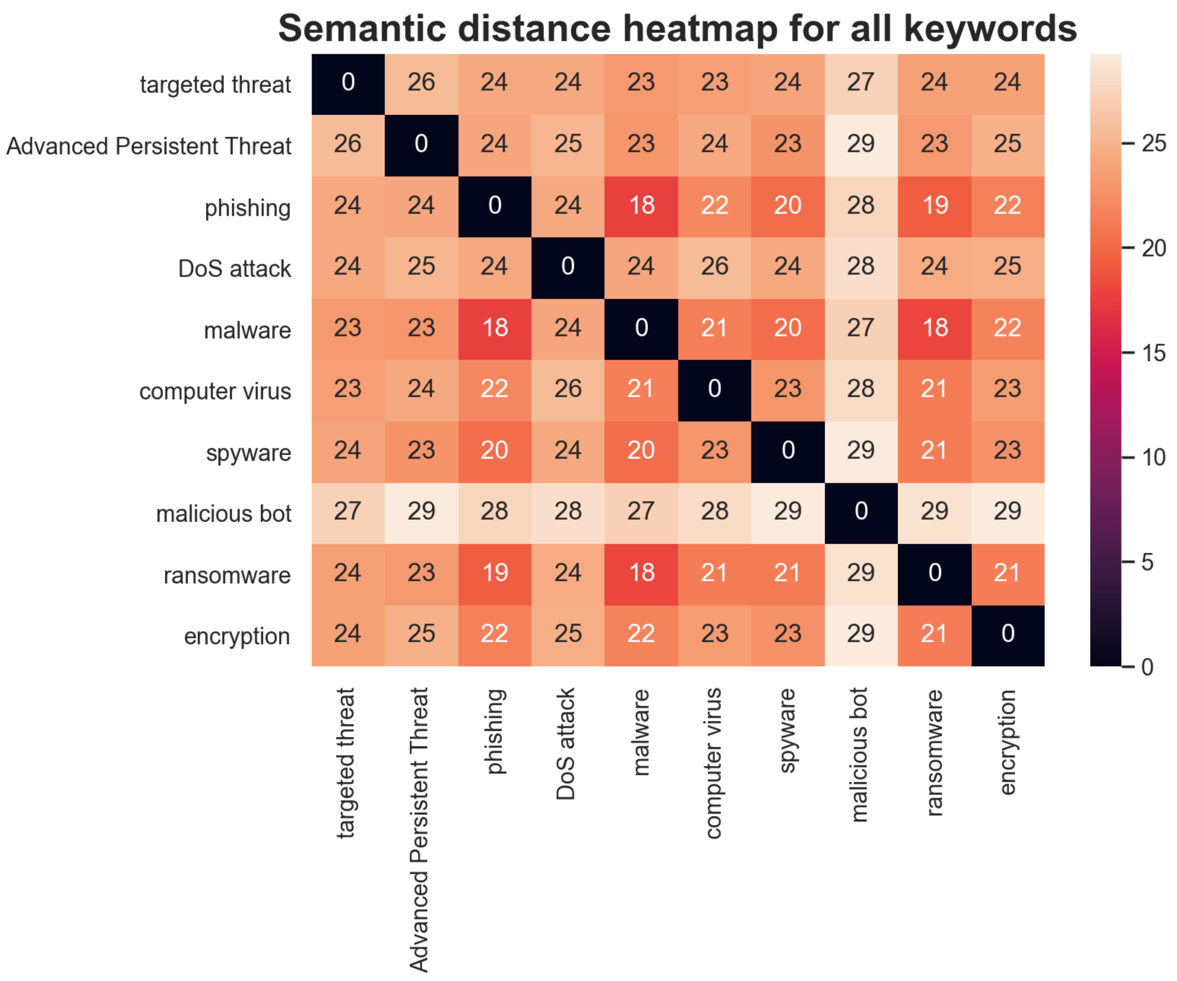


Figure 1: Heatmap with the distances between all keywords (darker means closer), to two significant figures. More precise results from the algorithm were exported to `distances.xlsx`

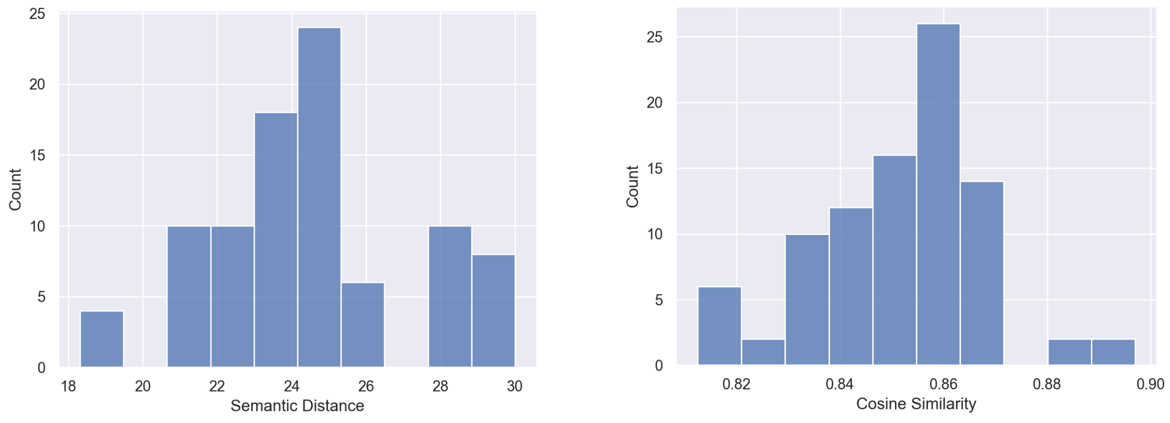


Figure 2: Histograms of the distribution of distances and cosine similarity scores calculated by the algorithm for all keywords.

It is possible to visualise and compare distances between keywords by studying the distance distributions of each individual keyword. This is achieved in Fig.3, which quantitatively and qualitatively reveals information about how far each keyword is from the rest. For example, 'malicious bot' has a relatively high mean and distribution of distances, whereas 'phishing' has the largest inter-quartile range of distances. All areas in the violin plot for Fig.3 have been scaled and equally normalised to prevent the number of articles downloaded from introducing any bias in the visual comparisons. Fig.4 visualises the relative spatial arrangement and embedded positions of the document vectors for the input keywords by converting the adjacency matrix from Fig. 1 to a connected graph. Although this visualisation relies on the same distance scores as Fig.1, the dimensional reduction from projecting a 100-dimensional vector space to a two-dimensional plot inevitably results in the loss of information, so Fig 3 is best interpreted qualitatively as the information distribution of keywords about their common centre of mass [3].

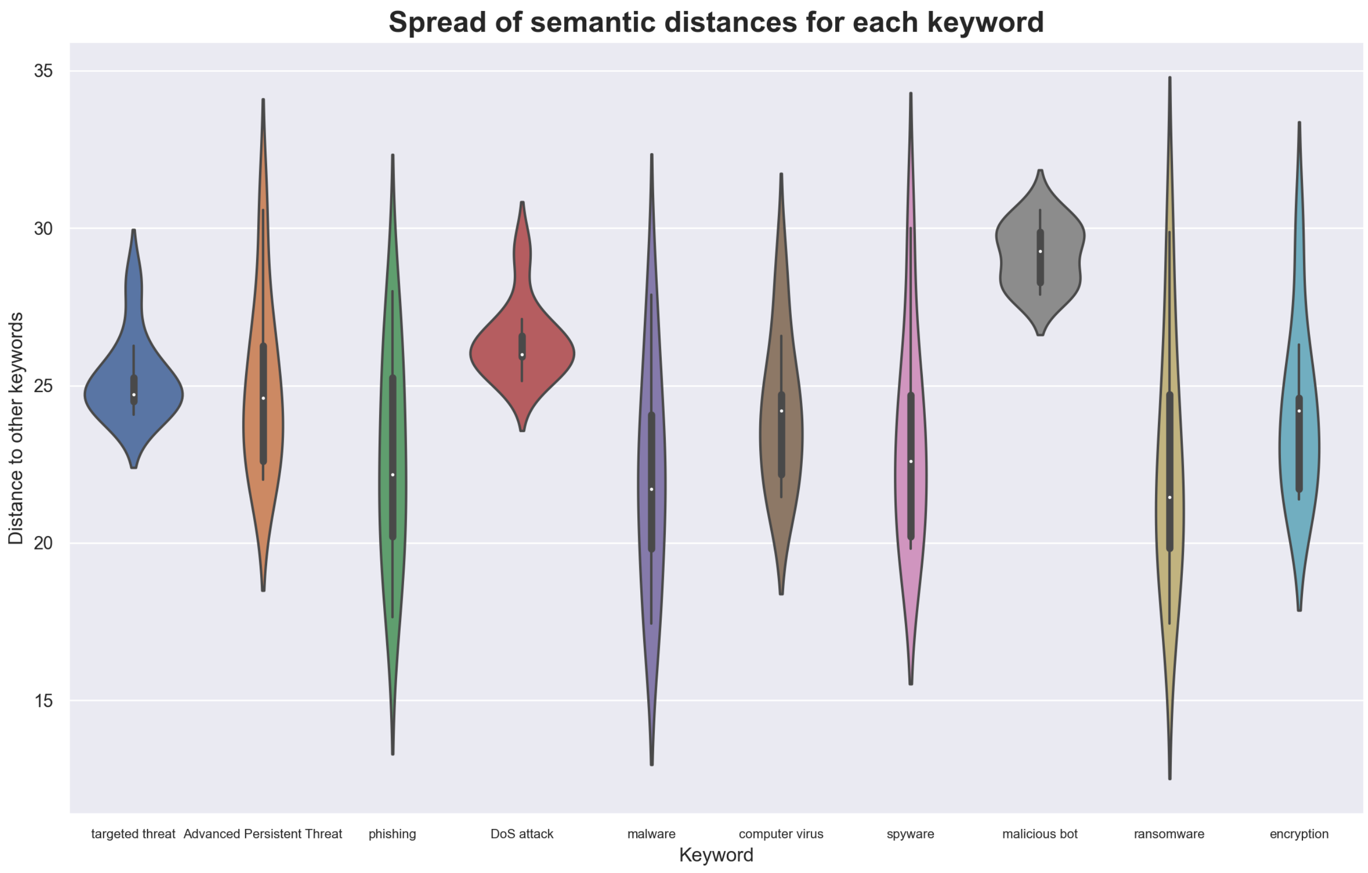


Figure 3: Violin plots for the distribution of distances to each keyword. Figure includes a boxplot with a kernel density estimate (KDE) plot for all distances to or from a given keyword's document vector.

Fig 5 has been additionally included to show the relationship between keywords in terms of TF-IDF scores. While a TF-IDF score is an indicative but not an accurate measure of semantic distance,it has been included to verify and double-check if the semantic distances calculated are in the right area. It was useful when tuning the neural network's parameters during experimentation.

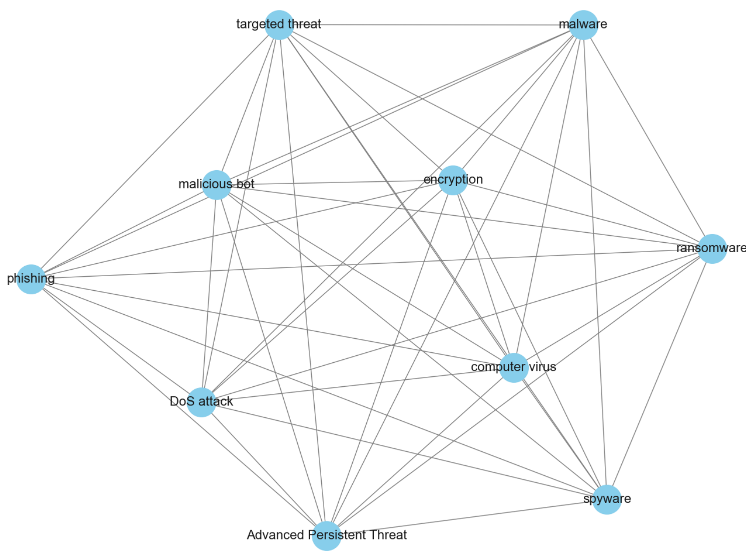
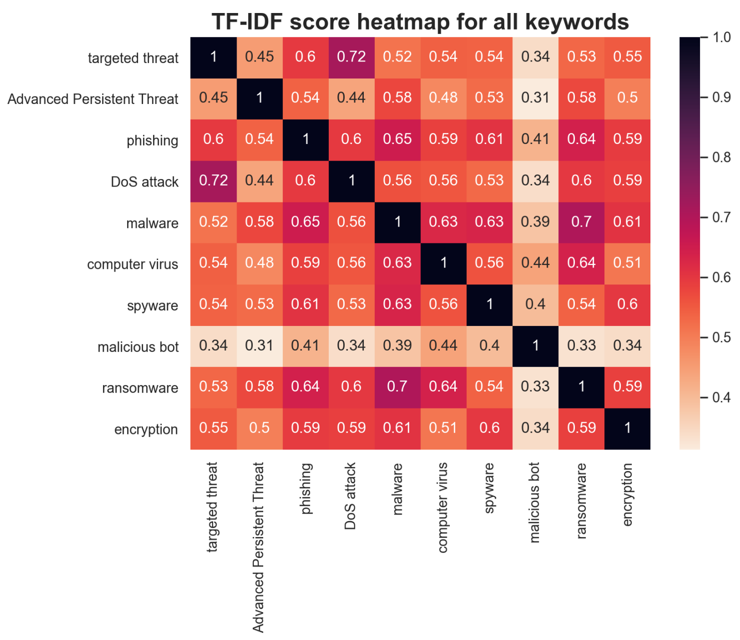


Figure 4: Graph representation of the semantic spatial embedding of the document vectors.



	targeted threat	Advanced Persistent Threat	phishing	DoS attack	malware	computer virus	spyware	malicious bot	ransomware	encryption
targeted threat	1	0.45	0.6	0.72	0.52	0.54	0.54	0.34	0.53	0.55
Advanced Persistent Threat	0.45	1	0.54	0.44	0.58	0.48	0.53	0.31	0.58	0.5
phishing	0.6	0.54	1	0.6	0.65	0.59	0.61	0.41	0.64	0.59
DoS attack	0.72	0.44	0.6	1	0.56	0.56	0.53	0.34	0.6	0.59
malware	0.52	0.58	0.65	0.56	1	0.63	0.63	0.39	0.7	0.61
computer virus	0.54	0.48	0.59	0.56	0.63	1	0.56	0.44	0.64	0.51
spyware	0.54	0.53	0.61	0.53	0.63	0.56	1	0.4	0.54	0.6
malicious bot	0.34	0.31	0.41	0.34	0.39	0.44	0.4	1	0.33	0.34
ransomware	0.53	0.58	0.64	0.6	0.7	0.64	0.54	0.33	1	0.59
encryption	0.55	0.5	0.59	0.59	0.61	0.51	0.6	0.34	0.59	1

Figure 5: Heatmap of TF-IDF scores to verify the semantic distances. Darker means more similar

Error Handling

1. In certain cases, the BBC's anti-bot mechanism blocks GET requests made by the program to the search page or article page. The algorithm includes exception handling that re-attempts a GET request after a brief sleep period if this happens. This is indicated to the user through print statements.
2. A warning messaged related to the absence of the `Levenshtein` library may occur, this was deliberately not installed since Levenshtein distances are not a measure of semantic distance.
3. For machines with a different character set to UTF-8 encoding, the user will be alerted during article downloads, but this will not affect distance calculations since the full article is in python data structures.

If, for any reason, the program does not complete or halts mid-run due to any unaccounted form of exception error, it would be extremely helpful if the program were re-run a second time.

References

[1] Q. Le and T. Mikolo, *Distributed Representations of Sentences and Documents*. Proceedings of Machine Learning Research vol 32(2), pp: 1188-1196 (2014)

[2] J. Nothman, H. Qin and R. Yurchak, *Stop Word Lists in Free Open-source Software Packages*. In Proc. Workshop for NLP Open Source Software (2018)

[3] B. Kievit-Kylar and M.N. Jones, *Visualizing Multiple Word Similarity Measures*. Behavior Research Methods vol 44, pp: 656–674 (2012)