Report – skbh77

# Problem 3

For any semantic distance algorithm, a word vector is required. I decided not to use the basic word frequency vector, because it would rate common words such as ‘a’ and ‘and’ very highly despite them not being directly related to the keyword of the article. Additionally, this type of vector would favour larger articles more than shorter ones. Taking this into account I decided to create the vectors using a term frequency – inverse document frequency vector (TF-IDF) vector. The idea behind this is to create a word vector for each individual word in each document which is more balanced than the straight frequency count. Initially, I create the base vector which is a dictionary containing the number of times each word is present within a document. Such as *“Encryption, [0,1,2,4,5,6,7,3,2,5]”*. Then, using this dictionary, the frequencies are altered so that it is logarithmically scaled, I used this because it means the more a word appears a document, the less important more occurrences of it become. The inverse document frequency is used to quantify the usefulness of a word, the rareness of each word across the documents. The ‘idf’ of each word is the logarithm of the amount of documents downloaded, divided by the amount of documents that word appears in. For example 10 Documents/’Encryption’ appearing in 5 would be log(10/5). To create the tf-idf vector for that word, each term frequency is multiplied by the idf of that word. This leads to a word vector for each word by keyword where a word is more important if it appears a lot within one keyword but is rarer in other keywords. This is useful for the purpose of semantic distance because if two keywords have large tf-idf scorings of the same word it would suggest that they are similar. Additionally, the scoring will place small significance on common words such as ‘a’ or ‘and’ that would naturally appear in all keywords. An example of the tf-idf scores produced can be seen in figure 1. Of note in the example is that ‘details’ which is a rarer word has a higher score for some keywords, whereas ‘of’ which is likely to appear is most keyword articles has a score of 0 because it is irrelevant to class it as similarity.



Figure - TF-IDF Example



Using this tf-idf vector I decided to use Cosine similarity to determine the distances. Cosine similarity was chosen because it will be more representative than just taking the Euclidean distance between the points and has been used within text matching and other similar applications. The algorithm is used for each combination of keywords, eg encryption and targeted threat. The top of the fraction is calculated as the sum of the product of the tf-idf values of each word. And the denominator is the square root of the sum of the squared tf-idf values of each word for one keyword multiplied by the other. This represents the formula specified in figure 2[[1]](#footnote-1). To retrieve the distance from this, the similarity is subtracted from 1. The similarities for each keyword are then printed to an excel file. This algorithm will produce results in the range of 0,1 which is useful because it means the visualisation can be set to have those bounds.

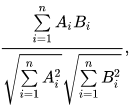


Figure - Cosine Similarity

1. Cosine similarity - Wikipedia. Available from: <https://en.wikipedia.org/wiki/Cosine\_similarity>. [April 20, 2021]. [↑](#footnote-ref-1)