Text Similarity Analysis

Summary and Purpose:
Text Similarity Analysis is a Natural Language Processing (NLP) and Network Analysis based technique to identify and analyze duplicate or similar texts. The algorithm is able to ingest thousands of digital media publications or social media posts. It can then group duplicate or similar texts based or create a social network of authors who published similar texts. These capabilities can help identify potential coordination between different authors.

Approach:
Our approach involves several steps:

1. *Create text representations*
   - After collecting the data (which can be composed of short social media posts or longer digital media publications), we generate a numerical representation of the texts in the form of real-valued vectors. There are two main techniques used for generating such text representations (which can be chosen by the analyst):
     a) *Bag-of-Words:* The bag-of-words approach represents a piece of text as a vector in which each value in the vector represents the number of occurrences of a word in that text. This approach disregards grammar, word order, and context and only tracks word frequency in a text. Although it is a simple approach, it works well for identifying text that contain identical or nearly identical pieces of texts because identical texts tend to have the same words occurring the same number of times.
     b) Word Embedding: The word embedding approach is a more complex technique that is able to generate a vector representation for words and is able to capture contextual relationship between words. Because this approach can capture context, it is more useful for identifying texts that contain words used in similar contexts or texts containing similar ideas/concepts (although may not necessarily contain identical texts) because vectors of words (aka word embeddings of words) used in similar contexts will be closer to each other in the embedding space.
   - Bag-of-words are generally better for identifying duplicate or very similar texts, but there could be instances in which word embeddings might be preferred as described above. For example, consider the following:
     - Sentence A: Russia is bombing Ukraine
     - Sentence B: Russia continues bombing the Ukraine region
     - Sentence C: Russia is attacking the Donbas region
     - Because bag-of-words measure the frequency of words, sentences A and B will have similar vector representations and hence a high similarity score than sentences A and C. If we use word embedding on the other hand, all three sentences will have really high similarity score because semantically they are talking about very similar concepts.
2. *Find similarity between texts*

  - Once we generate vector representations for our texts, we need to compare them. We utilize the cosine similarity measure to compare the similarity between our text representations (which are now high-dimensional vectors). Among other reasons, the cosine similarity measure is better than other measures used for vector similarity because of its efficiency on sparse and high-dimensional vectors.
  - We often use a thresholding parameter (between 0 and 1, chosen by analyst) to efficiently identify texts that we deem are similar. This helps get speed up the algorithm and reduce noise.
3. *Group texts based on similarity*
  - Once we have our cosine similarity scores, we group our texts based on these scores. For example, if documents A and B both have a high cosine similarity score to document C (i.e., A and B both very similar to C), then all three documents will be in the same group. An analyst can look into these groups for further analysis, e.g., identify what texts are being duplicated, who is duplicating them, and to what extent.
4. *Create an edgelist between authors publishing similar texts*
  - We can also use the above results to generate an edgelist—a two-column matrix representation of a network graph as a list of its edges. The analyst can specify the entity to use as nodes. For example, the common choice for nodes in a graph is the author of a text. This means the network will be composed of nodes (authors) where an edge (connection) between two nodes (authors) means that these two authors published similar texts. The edge weight (numerical value assigned to the edge/connection) would represent the number of texts that were similar between the authors. There are several way we can threshold the network to reduce noise.

Limitations:
Our main limitation is computation speed and memory. Because we need to compare the text representation of one text with all other texts, calculating the cosine similarity (step 2) can often take a long time and require a lot of memory. Even though we have made several modifications to increase the efficiency of the algorithm, we are still limited by hardware capabilities. We hope to continue improving our algorithm and perhaps utilize cloud capabilities in the future to overcome hardware bottlenecks.