3 Bonus

(40 Points)
Please complete Part 1 and submit for grading before starting the bonus (i.e., this part). Without Part 1 completed, this will not be graded. The questions must be completed in order (i.e., no skipping ahead to questions of choice). There will be an independent Gradescope for this, with no autograder, so please be sure to treat this as an extension of Project 2. No exceptions.

For bonus, we will first look at and analyze the metrics; then, we will do the same for the embeddings.

3.1 Bonus Summary and Rubric

For this part, do the following:

a. (*10 pts.*) Add functions called euclidean_distance and manhattan_distance. Then, process the documents using the new metric, analyze and discuss the differences between these and cosine similarity.

b. (*15 pts.*) Add a function called bm25 and implement as descibed in Section 3.3. Then, again, look at the top 20 words for document 1.txt, and then create a cosine similarity matrix and compare it to the one generated for Part 1.

c. (*15 pts.*) Add a function called sbert and implement as descibed in Section 3.4. Then, again, look at the top 20 words for document 1.txt, and then create a cosine similarity matrix and compare it to the one generated for Part 1 and using bm25.

3.2 Distance Metrics

Implement euclidean_distance and manhattan_distance to find the L2 and L1 distances, respectively, using the same interface used for cosine similarity. From the results, and via outside resources, discuss the differences between the three metrics. Use sample results to demonstrate the concept shared whereever possible.

3.3 BM25

TF-IDF effectively normalizes ”useless” words, as discussed in the previous section. However, let us look at a shortcoming: scaling happens linearly. In other words, given a document that contains the word “dog” twenty times, and another from that corpus that contains that word ten times, the former would be double the magnitude of the latter. Although “dog” might be more prevelant in that first document, the other is likely closer to half of its relevance. BM25 normalizes TF-IDF for such cases. The implementation is as such.

$$\frac{\sum_{i=1}^{k} TF(w, F_k) * (q+1)}{|F_k| + q(1 - b + b * \frac{|F_k|}{F_{avg}})} * log(\frac{|C| - |F \in C : w \in F| + 0.5}{|F \in C : w \in F| + 0.5} + 1)$$

,
where constants take on the following values: $q = 1.25$ and $b = 0.75$, and $|F_k|$ is the number of words (i.e., size) of document k and F_{avg} is the average size considering all documents in the corpus C.

3.4 SBERT

As discussed in class, the aforemented content-based embeddings lose spatial relationships between words. Hence, semantics are not captured when these sparse representations are used. Let's try a dense embedding and see how things change.
One of the Large Language Models powering chatbots like ChatGPT are Bert-based. Let's use a variant of BERT to determine the similarity in documents called SBERT. For this we will use a python package [https://www.sbert.net/].

```
pip install -U sentence-transformers
```