Obtaining Sentence Embeddings from the “bert-base-swedish-cased” pre-trained model.

1. Removing stopwords and punctuations for data pre-processing
   1. No further pre-processing should be done because the bert model is trained on raw text. Doing further pre-processing such as lemmatisation or stemming may cause the removal of some context of the article, given that a single word can have multiple meanings and ways of representing a context.
      1. One can argue that lemmatisation should always be done in our case of finding similarity, as it increases the frequency of content words that we are interested in. For example if the article is relevant to new inventions and there are words like “invent”, “invents”, “inventing”, “invention”, “inventions”, “invented”, then ideally its better if they were all represented in the same manner, rather than treating them as different words.
      2. However, based on the fact that the bert model trained on raw text rather than lemmatised or stemmed text, we can already expect the different variants of the word vector to be close to each other in general. As they are still different word vectors altogether, the difference in their positions in the vector space can be explained by the context in which each different form of the word is used. As such, performing further pre-processing removes the possibilities of the different variants of a word, which affects our sentence embedding altogether.
   2. Stopwords were removed because of the purpose of this project, which is to find similarities between the articles. If the stopwords were not removed, what would happen is that the sentence embedding will be overwhelmed with the weights of the stopword. This brings each sentence embedding much closer to one another and more similar to each other, which ultimately affects the accuracy and precision of differentiating the articles that are not supposed to be relevant to each other.
2. Tokenise the articles using the same tokenizer that the pre-trained bert model used as a necessary step.
3. Using the tokenised data, produce the word embedding for each word in the article
4. Obtain the sentence embedding by taking the mean of all word embeddings in the article
   1. Here you can see the reason why stopwords were removed
   2. As stopwords are frequently used in a text, and the sentence embedding takes the mean of all word embeddings, then the stopword vectors hold a heavy weight in the sentence embedding. As a result, all sentence embeddings of the articles become closer to one another when the stopwords are not representative of similarity.
   3. This ultimately affects the accuracy and precision of differentiating the articles that are not supposed to be relevant to each other.
5. Obtain cosine similarity between each articles using sentence embeddings. The higher the cosine similarity score, the more similar the pair of articles.
6. As evaluation, the average percentile of the averaged cosine similarity scores of articles with the same tag is obtained
   1. Sort Tag by frequencies
   2. For each tag, average the cosine similarity scores amongst articles with the same tag
   3. Obtain percentile as compared to the cosine similarity scores for the rest of the articles
   4. Repeat process for every tag
   5. Average the percentiles obtained for every tag
      1. Result: 93.33 percentile
      2. Optimistic result as having a high percentile is indicative of a good performance as it shows that articles with the same tag generally have a cosine similarity score distinctly higher than articles that does not have the same tag.