VIMEO CODING PROJECT (Shruti Kannan, kshruti@bu.edu)
Github link: https://github.com/kannanshruti/video_recommendation

Aim:
To accept a video clip ID as a command line argument, and return 10 most similar videos

Brief outline of algorithm:
1. The solution, first, extracts all relevant information from the 3 input csv files, and combines it in a single 'DataFrame' for easy access. In order to include the maximum possible data, only rows containing 'NaN's in the clip 'title' were excluded.
2. Next, words are extracted from the clip 'title' and clip 'category names' by default, and this serves as input to the word embedding model. Stemming was also performed on the words to increase the chances of finding similar words across the dataset, and to remove unnecessarily repeated words in the model vocabulary. Options are provided for including 'category names', 'captions' and excluding 'stop words' from this input 'human' vocabulary.
3. Next, the model is now created using these words, with minimum occurrences of each word as 1 and the dimension of the resulting vector for each word is 10.
4. For each 'clip ID', the associated words in the 'human vocabulary' are now replaced by the corresponding word vectors that the model computed. Since the number of words associated with each 'clip ID' are different, the average vector for each 'clip ID' is computed, and used as its representative.
5. For each query ('clip ID' for which recommendations are sought), cosine similarity is used to measure 'similarity' (given by high cosine value), and the top 10 of them are returned as result.
6. Each 'clip ID' is associated with a corresponding JSON object which has the relevant clip information. Each of the resulting similar videos from the previous step is now replaced by its corresponding JSON object addressable by the query 'clip ID' as key, and this result is converted to a JSON list and returned as the final output.

Challenges:
1. Captions: Since most videos have very long captions, including captions caused the program to hang. This has been included as an optional argument in the function which extracts words for each sentence ('get_sentences'), and can be included when a faster computer is used.
2. Stop words: Some 'clip titles' were just stop words (e.g. 'There Again'), and without any given 'categories', hence excluding stopwords while not including 'captions' would have resulted in empty vocabulary vectors. Hence stopwords were not excluded.
3. Minimum word occurrences in vocabulary: This option can be used to ignore words which are rare in the dataset, and hence may skew the 'average vector' computed for each 'clip ID'. This option was, however, not used in this solution (i.e. it was set to 1), since a lot of words had few occurrences and with captions not being included, this resulted in some cases having no remaining words, thereby, decreasing the pool of 'clip ID's. Once 'captions' are included (more number of words), this option can be increased to 5, to remove the effect of rare words on the 'average vector' for each 'clip ID'.
4. Some data was not in English, but the number of such cases for each foreign language is low, and the model probably does not function well in those cases.

How would I build this for real at Vimeo, and performance considerations:
Although this code can be scaled to a larger dataset, just by itself, it may be very slow (considering how much time a user would be willing to wait to get recommended videos is, about a few seconds) for large datasets. This model can be included as part of a neural network which would also increase accuracy. Using too many words could lead to the average clip vectors being incorrect. Rare and stop words should be excluded. Also, clips with non-English information need to be better handled.

Additional techniques I would leverage to improve the results:
1. Stemming and removal of stop words to remove unnecessary words from the model vocabulary.
2. Include this model as a layer in a neural network.