Objective

Assess theoretical & practical differences between the current initial model implemented that uses GloVe word-level embeddings, bag-of-words & soft-cosine-similarity and the proposed Universal Sentence Encoder (USE) model.

What are the models used for?

The models are used to convert text-documents to vector representations. These vector representations allow us to compare documents quantitatively in terms of their similarity. One use case for example is that given a document A, find the most similar documents to A from the corpus. Our corpus has more than 600,000 documents hence why we require algorithms to analyze and compare these documents as opposed to a human-reader manually evaluating each document.

Current Model & System

The current system works as follows:

1. User inputs a search query. E.g. user inputs the keyword “password”.
2. MongoDB find function is used to find a subset S of the 600,000 documents that contain the keyword “password” (this is fuzzy in nature i.e. the word “password” is matched both exactly and non-exactly)
3. Documents in the subset are ranked by using Soft Cosine Similarity(SCS) on GloVe embeddings & bag of words representation of the documents. The SCS compares the query to each document.
4. If a user “likes” a document, we can compute the SCS of all the other docs with the liked docs to re-rank the results.

    Evaluating the Soft Cosine Similarity + GloVe + Bag of Words (Call this System A)

    Bag Of Words (when used with term frequency vectors) essentially extracts all the unique words across the whole corpus and then for each document builds a word-count vector (aka term frequency vector).

    For example:

    After preprocessing the sentences, each sentence will be converted to the following bag of words representation (key = word in that sentence, value=count of that word in that sentence):

    Sentence 1 = {"John":1,"likes":2,"to":1,"watch":1,"movies":2,"Mary":1,"too":1};
    Sentence 2 = {"Mary":1,"also":1,"likes":1,"to":1,"watch":1,"football":1,"games":1};

    Then, both of the dictionaries are combined to have 1 master list of unique words.

The master list will look like:
Master Word List =
{"John":1,"likes":3,"to":2,"watch":2,"movies":2,"Mary":2,"too":1,"also":1,"football":1,"games":1};

Then, each of sentence 1 & 2 will be represented in terms of the master list but the counts will be replaced by the counts of the words in that specific sentence. For example:

Sentence 1 term frequency vector = [1, 2, 1, 1, 2, 1, 1, 0, 0, 0]
Sentence 1 term frequency vector = [0, 1, 1, 1, 0, 1, 0, 1, 1, 1]

First value in the above vector for sentence 1 is to be read as “the word John appears 1 time in sentence 1” and first value for sentence 2 is to be read as “the word John appears 0 times in sentence 2”.

Note that no NLP model has been used yet.

GloVe is a NLP model that converts a word to a fixed-size vector. Words that are similar, will be closer together in the vector space.

Although GloVe is a word level model, in System A it is being used for document classification. To achieve this, we need to use Soft Cosine similarity combined the term frequency vector combined with the GloVe word embeddings.

The GloVe model is just used to convert all the unique words to their respective vectors. Then, if we have M words, an MxM matrix is built which computes the similarity between each word.

Now we have firstly our term frequency vector representing our documents, and our similarity matrix representing similarity between individual words.

To compare documents we can either directly use the term frequency vectors or we can use the latter with the word similarity matrix.

If we only use the term frequency vectors to compare documents, we will just compute the similarity between each documents’ term frequency vector by using some distance/similarity measure like cosine similarity. Assume we use cosine similarity.

In this case the problem is that two documents may have 0 overlapping words. Then, the cosine similarity between them will be = 0. This gives extreme results because for example if the word “play” appears exclusively in doc A and “player” occurs exclusively in doc B, then taking the cosine similarity of A & B will return 0 suggesting that these

docs are not similar at all… however we know that “play” & “player” are highly related words. This is where the GloVe Similarity matrix comes in.
Soft Cosine Similarity allows us to compare the term frequency vectors with each other while also keeping the word similarity in mind.

Universal Sentence Encoder (USE) Based System (System B)

The USE based system, works as follows:

1. User inputs a search query. E.g. user inputs the keyword “password”.
2. MongoDB find function is used to find a subset S of the 600,000 documents that contain the keyword “password” (this is fuzzy in nature i.e. the word “password” is matched both exactly and non-exactly)
3. If a user “likes” a document, the USE model is used to convert each doc to a fixed size vector (analogous to GloVe but sentence-level instead of word level) and also the initial query is converted to the same sized vector. The query vector is moved in the direction that is towards the liked documents’ vector in N-dimensional vector space where N is defined by the USE model.
4. Then, the cosine-similarity is used to calculate similarity between the updated query vector and all documents. (SCS not required).

System A vs System B

	System A	**System B**
Model Used	Universal Sentence Encoder	GloVe + SCS
Models Pros	Sentence Level (potentially captures semantics better). No Preprocessing required. Handles Out Of Vocab words. Available in both O(n) and O(n^2) variants where n is sentence length. Whole sentence is looked at when generating vector.	Optimized for vector space operations such as King - Man = Queen Potentially Faster than USE in some cases
Model Cons	High accuracy model has O(n^2) runtime	Word context is defined manually by authors (not all

		neighbouring words are looked at when generating word vector) Word order isnt taken into account. Can’t handle OOV words. Requires SCS since it is word level. (this is huge). Not sure if can be used for iterative vector update. Large documents’ similarity score will be noisy due to SCS and word level features.