Shea Durgin

https://github.com/sheaDurgin/NLP23/tree/main/Assignment_3

Part 1: My results came out to be ~11% when using question titles and ~3.5% when using question bodies. So I found a 3x increase in performance when using titles. To me this actually isn't surprising. Question bodies on forums such as stackexchange can be very long and this could perhaps bring in a lot of noise to the content. The question titles are usually quite short and to the point. Thus similar questions have a higher chance of having the same exact title rather than body. This is because they write more in the body, thus more chances for variation. What I find interesting is that for humans, the body text will bring more value to us since there is more information and context. However, for the computers (on this basic model), this extra information can end up being noise and interfering in its process of finding similar questions. Another thing to consider is that cosine similarity uses text length as a parameter, and similar questions are not guaranteed to have a similar text length. Thus, a more advanced model should be used for this problem (hence part 2).

Part 2: My results came out to around 60-70%, sometimes it would dip and sometimes it would go up but it was usually in the 60s and 70s. My method for doing this part was to concatenate the vectors that were positive samples and negative samples and train the model by telling it which ones were a positive sample and which was a negative sample. There was an equal number of positive and negative samples to hopefully stop the model from spamming one guess like I ran into with naive bayes earlier on in the class. The model was a simple feedforward neural network without any bells and whistles. Overall the model isn't that great, while it is way better than the fasttext cosine similarity version, it is still barely better than flipping a coin. I feel like a potential improvement would be a larger dataset of positive and negative samples. There are only ~280 each which in terms of natural language processing is quite small. I also wonder if my work even had any effect as a fellow student told me they had similar results without even properly being able to train their model.

What are the differences between Skipgram and Continuous bag of words approaches? Describe the advantages and disadvantages of each.

Skip-gram and Continuous Bag of Words (CBOW) are two different algorithms for training word embeddings. Both approaches are based on the idea of predicting words in a context window, but they differ in how they use this information.

CBOW is a model that predicts the probability of a word given its surrounding context words within a window of fixed size. It takes a sequence of input words and tries to predict the center word of the sequence by summing up the embeddings of the context words. CBOW is trained to minimize the average negative log-likelihood of predicting the center word given its context.

One of the main advantages of CBOW is its computational efficiency. It operates by summing up the embeddings of the surrounding words to predict the target word, which makes it faster to train than other models. Additionally, CBOW can capture the overall meaning of a sentence or

document by taking into account the co-occurrence of words. However, CBOW may not perform well for rare or infrequent word contexts, as the embeddings for these words might not be well-trained. Additionally, CBOW treats all context words equally, which can make it difficult to capture the nuances of word meanings, as it doesn't differentiate between them.

Skip-gram is a model that predicts the surrounding context words given a center word. It takes a center word and predicts the probability of other words in its context window. Skip-gram is trained to minimize the average negative log-likelihood of predicting the context words given the center word.

One of the main advantages of Skip-gram is its ability to capture the meaning of rare words or infrequent word contexts. This is because it trains embeddings for each word in multiple contexts, which makes it better at handling rare words. Additionally, Skip-gram is more flexible and can capture the nuances of word meanings since it treats each context word separately. Then there are the negatives, Skip-gram is computationally more expensive than CBOW because it trains embeddings for each word in multiple contexts. Moreover, Skip-gram might not be able to capture the overall meaning of a sentence or document since it doesn't take into account the co-occurrence of words.

In summary, CBOW is computationally efficient and useful for capturing the overall meaning of a sentence or document, while Skip-gram is better at capturing the meaning of rare words or infrequent word contexts and is more flexible in capturing the nuances of word meanings. The choice between the two approaches depends on the specific task and the characteristics of the data being used.