Part 4:

a) Our vectors are stored in a hashmap with keys of every word from the indexed file and values of each word's semantic description vector. The asymptotic memory usage for one semantic description vector is O(n) where n is the number of words in a sentence because the vector stores a key value pair for every other word in the sentence excluding the key itself. The memory usage for all of the vectors is O(s * n) where n is the total number of words from each of our sentences and s is the number of words in an individual sentence; because the master hashmap stores the vectors and their pertaining words which is O(n) and each vector stores a key value pair for each of the other words in the sentence excluding its own key word which is O(s), resulting in a O(s * n) memory usage. This memory usage is reasonable because we're not representing any unnecessary information such as storing key value pairs for the total words for each semantic description vector.

b) The algorithm for cosine similarity involves concatenating two key sets for each of the vectors passed into the function and using this new set to calculate the numerator, then using the individual key sets to calculate the denominators for each vector. Then we divide the numerator by the square of the two denominators multiplied together. The running time for the cosine similarity function is O(n) where n is the total amount of keys from the concatenation of the two key sets from the vectors. Yes, this running time is reasonable because we do the minimal amount of calculation that is necessary to find the cosine similarity for every vector in the master hashmap and it runs in an efficient amount of time.

c) The algorithm for our TopJ function is to calculate the cosine similarity for the passed in word and every other vector in our master hashmap and then store the similarity in a pair containing its given word and the number for the similarity. We then sort the pairs by their values from greatest to least and then return the number of requested pairs. The running time of our overall TopJ function is $O(n^2)$ where n is the number of unique words in the text file because we calculate the cosine similarity which is itself O(n) for the passed in word compared to every other vector in our master hashmap. Yes, this running time is reasonable because we need to calculate the cosine similarity for the word's vector compared to every other vector in order to store the similarities in their respective pairs and then return however many pairs are needed.

d) We reduced the running time by not storing a key value pair in the semantic description vectors for every single word from our sentences. This changed the memory usage of our master hashmap from $O(n^2)$ to O(s * n). This resulted in a more efficient representation of every vector for all of the words in our sentences because we eliminated storing unnecessary information.

Part 5:

The TopJ query does get better with more data because as we increase the amount of vectors we get a more precise answer. This is because if we have a jump from 1 to 2 there is a big change in distance; however, if we have a change from 199 to 200 there isn't as big of a difference.

For the first book and second book, if we run a cosine similarity on the word snap, we get pairs such as [comrad=0.9545454545454546, niggardli=0.9545454545454546, finger=0.9545454545454546, itiner=0.8313979615881406, dare=0.6908492797077574]. If we run a negative Euclidian distance we only get 0.0 values for each word in the TopJ query and if we run a negative Euclidian normal distance we get pairs such as [old=-0.7857354069993928, upon=-0.768924377627793, van=-0.7602600281404731, turn=-0.7452576066853085, everi=-0.7251272807722348]. The cosine similarity and negative Euclidian normal distance have somewhat similar distance values for their TopJ queries; however, the negative Euclidian distance only has 0 values because for every key value pair in the semantic similarity vector it will have a 1 value and the norm of 1 minus the norm of 1 will result in 0 every time. There is not enough data for the negative Euclidian distance to be effective in telling us the similarity based on just one book and only 1 values. However, the cosine similarity and negative Euclidian normal distance give reasonable results of what the similarity is between two vectors.