My design was mostly to handle the queries by having the index returning all possible locations of the given query then handling the results separately. While this is sufficient for this project it leads to a bit of unnecessary code, for example I have no way to handle the union of multiple different queries without an external handling of it. Likely for the next project and scoring, I will need to convert my union to a general union. To handle the searching of phrases, I find the intersection of the posting lists where the word is i spots away from the first. So the ith word in the phrase would be the location of the first word + i. My implementation closely matches the implementation given in the class where you go to the largest docid because there can be no intersection on the smaller docid pages. A larger query would be handled the same. The hardest part I had with the project was trying to keep my code as efficient as possible. I was trying to avoid just excessive loops over the list which was mitigated by things like skipping to the largest doc Id.

The main software libraries used are data structures, file management. To begin with the data we were given was in a compressed json file. So an assortment of Java standard libraries were used to open the file. Then to parse the json file I used the simple-json library provided to us. For data structures I used an assortment of different types. Maps were used for most of the primary data storage and the backbone of the inverted index. For the postings I used a linked list in order to keep all occurrences of a word in a document.

Features might be misleading for comparing different scenes because plays are not static. For example in certain plays the characters may move from England to Denmark in different scenes. So the frequency of Denmark in one scene might be 0, and irrelevant, but in the next scene is very relevant.

The queries that took the longest to complete were the multi word queries with words that are common. The fastest were one word queries. This is due to the fact that the multi word queries not only have to search multiple words, but then also have to check all the locations of the found word. As such phrases with words with multiple occurrences, that are like “let slip” take the longest.

The shortest scene was antony_and_cleopatra:2.8, with the longest being loves_labors_lost:4.1, with an average scene length of 1200 words.