CS4417 Assignment 2 Report

Yixin Jiang

# Describe the design you used for Part 1 and 3.  This should include the keys and values you used.

In part 1, I used city names as keys and the number of Starbucks in each city as values. I used the String.upper() function to get rid of any inconsistencies in capitalization. One thing I could have done differently is check the data for more inconsistencies, such as spelling errors or different ways of abbreviating cities (if applicable).

In part 3, I used genres as keys and movie names as values. A noticeable thing I did was separate everything in my reducer output with tabs. The format was this: genre \t movie1 \t movie2 \t movie3 \t … and so on and so forth. I did this so that strip would get rid of any unwanted blank entries when I inevitably split the list of movies later on, however, separating the genre and each of the different movies with the same sequence (the tab) might have confused other users later on.

# What else would you have done in the inverted index implementation, given more time, energy, resources, etc.?

I would have implemented the ability to use both AND and OR and to have multiple terms. Currently, my implantation is limited to a maximum of one Boolean operator and 2 search terms. I would also have kept track of the number of occurrences of each key. This way, I can optimize order for query processing, so I can process in order of increasing frequency.

# How difficult was it to implement the inverted index? How difficult would it be to implement another task, given this experience? What would be straightforward? What would take more time?

Once I understood the concept of the inverted index and the context of the assignment at hand, I realized I could use Python sets, which contain built-in functions that simplify Boolean operations. Given this experience, the most difficult part about implementing data structures is understanding the fundamentals and the reasoning behind every decision. A straightforward task would involve something similar; if it is taught in class and does not require complicated and obscure algorithms, it should be doable given that I attended class. A more time-consuming implementation might be one that requires libraries I do not know about, or that I write my own classes and functions.