Project 3 Logan Docherty

Additional Cleaning: I did have to do additional cleaning on the beginning data because I found that we had a mass amount of duplicates within each table. This was primarily due to one of the web pages ending up repeating the same books after 100 pages.

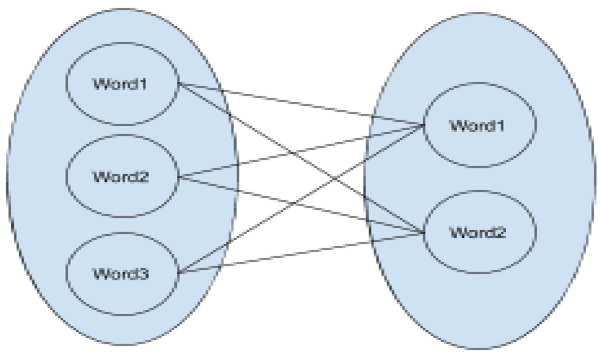
Due to the removal of duplicates we had a smaller data set for Table A. Overall, we didn't see as much similarity as I would have liked from the two tables. Although, this is a tougher dataset because of the keyword 'Children'. The data was both collected by searching the Children's genre on their respective page. Therefore a large amount of the data container "Children" in the title.

Problems: There were a few problems I encountered during this stage. Primarily, I had to readjust the threshold multiple times. I originally set the threshold to 0.9 (ie 90%), while this seems like it would capture all our potential matches, this matched some titles that should not have been matched. Ex: bambi and babel. When I pushed this threshold to 0.95, we saw much greater success. The only other problem was run time which was expected to be long but could definity be improved by pruning in the future.

Table A Size	**Table B Size**	**Cartesian Product(AxB)**	**Table C Size**
1849	3538	6,541,763	37

Generalized Jaccard explained:
I used the Generalized Jaccard algorithm to do matching on this set of data. After splitting the string titles(i.e tokenizing), we pass this array of strings into Generalized Jaccard. This algorithm then gives a similarity score to each pair in the passed titles array. The algorithm then only allows the pairs that meet a certain score threshold to pass through.
Then we form a bipartite graph, such as **figure 1.** Each line in figure one would have the score of how much the two words matched. We then take the largest scores from the pairing nodes. Each node can only end up with one score. We then add these scores and divide it by the number of words(number of words in group 1 + number of words in group 2) + the maximum-weight matched in the graph. The algorithm then returns the normalized(0-1) output.

Figure 1:


Word1
Word2
Word3
Word1
Word2