Justin Do
CSC483

Final Program Report

In this project, I went ahead and attempted to try a multitude of ideas (along the given guidelines) and concepts to see which yielded the best results. To begin, I started by indexing the given documents 3 times as requested. Once for no stemming or lemmatization, once for lemmatization, and once for stemming. I would've liked to have tried both stemming and lemmatization, but having just seen the awful amount of time it took to build the lemmatized index, I refrained from that. For all of the indexes, I did my best attempt to scrub any possible strange Wikipedia formatting. This included the removal of brackets per new document, the removal of "==" from sections/topics, bad input results that may have been a one off thing (like a line just containing "["), and other weird things that didn't follow the format stated, like files and images being listed on new lines as [[File: or [[Image:. I do think some poor inputs probably slipped through but it's quite hard to know without having read every single piece of text that comes by. Also, there was tons of foreign keys and characters that I simply could not modify/handle. As for handling the text itself, I let the NLP library handle them for the different indexes. I didn't remove any stop words or anything like that, I figured that the scoring algorithm would deal with them pretty fairly. For querying, the first thing I wanted to do was to try our having the query in bigrams. This turned out to be… utter garbage. What I would do, is that I'd make every combination of bigram, query them into an index, get the top 10 results, and keep track of those results by having a running total score, and how many times it's shown up. I tried analyzing them by both highest running total score, and highest average score (where I divided the running score by the number of documents that were added. It seems because of stop words and the fact that bigrams need it to have the specific phrase in there that stopped this from

working so well. I wanted to do a stop word removal for the query but I decided against it in the interest of time. I also considered just removing the first few words of the query as most of the time they'd be stop words or nonrelevant. I then just decided to go with the standard full query into the indexes. This actually worked better, but still not great. I tried both using and not using categorizes for the query search, and to my surprise, it made a pretty big difference. You can see the results below.

It should be at this point that I was measuring them with MRR and somewhat with P@1. MRR was used because we clearly care about a single correct answer and that it is the first answer that is given to us. In Jeopardy, we are only allowed to give one answer that is or isn't correct. Thus, we need for our program to return us the right answer in the first slot every time, otherwise we'd be wrong. If this was something like Family Feud where there could be multiple answers, then sure we probably wouldn't want this, but it's clearly not. In using MRR, we also get to measure if any tuning to the scoring or ideas increases the correct document's position in their respective lists. P@1 could've also been used, albeit it is a lot more all or nothing, which does work, but it may end up being hard to see if minor adjustments have any impact unless it consistently just brings the precision up. It's also can be very volatile depending on the set of queries given. Too small a sample size could definitely lead to bad numbers or overly good numbers. MRR allows me to get an average idea on how often it is right. I also only considered the top 10 for MRR, after all, who looks on the second page of Google? My results for each (using BM25 and without category) were:

No lem/stem: `MRR: 0.21467857142857139 P@1: 15`

Lem: `MRR: 0.21453968253968247 P@1: 15`

Stem: `MRR: 0.2130119047619047 P@1: 15`

But with Categorizes I got:

No lem/stem: `MRR: 0.25325793650793654 P@1: 19`

Lem: `MRR: 0.24675396825396823 P@1: 18`

Stem: `MRR: 0.25286507936507935 P@1: 19`

Surprisingly, there was little difference in no lem/stem compared to both lem and stem in the "no category" section. I'm fairly certain I have implemented properly, but I guess that's just how it is. In fact, lemmatizing and stemming it did make it a bit lower. I assume that due to how the querying was handled, not having those exact words make lemmatization and stemming lose some points, but not a significant amount. I should've tested stemming the query. Anyhow, with categorizes, a relatively large difference was made. Nearly 1 percent and 1 more P@1 is impressive. Again I assume it's due to how the query was being handled and not adjusting the query to match the type of index it was going against.

In changing the scoring function, I experienced a massive drop. For all three of my indexes, my MRR dropped to sub 0.01 for all 3 and I received 0 P@1 across the board. I can't exactly understand why this is the case other than maybe the document just happening to not show up in the top 10 and that more documents were relevant this time around

So my best system is actually the no lem/stem. I think that there's only so many times that an "impactful" words in a question that the amount of pages you can pull up are rather limited. For instance, when I added the category section to my query and when I didn't. The category was clearly "impactful" and changed the way my documents ended up being scored. As for the questions I got incorrectly (as in outside the top 10), I noticed that many of them involve

numbers when compared to the ones I do get right. Just about every one of the incorrect ones have some form of number. I suspect it performing so poor is due to the vast amount of documents that could contain these numbers. Maybe the scarcity should help scorer in finding it but this doesn't seem to outweigh the amount of documents that may contain the number. The other group I noticed that caused lots of incorrect errors, are the queries that are either too short or too long. Queries that are too short are usually under 4 words or 5 if it contains a few stop words. Queries that are too long can span over 8 or 9 words. Both of these were to be expected as the longer the query, the harder it is to know which words really matter (even if we weighted it, things like stop words could appear so often that it "carries" a document, and the smaller it is, the more vague and the less we have to work with to score.

I've explained pretty well why I think the no lem/stem combination worked better throughout the paper but to summarize it probably comes down to the query not being lem or stem and thus probably reduces the scores of the searcher when it comes across words that don't "match" despite the words being synonymous or a form of the word, but not by a significant margin of course.