COMP 479 - Project 4

François LaBerge

December 10, 2020

1 Different Behaviours of Ranking Schemes

2 Issues with the tf ranked postings list

No issues complicated issues were encountered while implementing the tf-ranked postings list. To keep the lists ranked during the indexing process we use python "heapq" library to create a priority queue that will keep our list sorted throughout the indexing process. The algorithm on page 45 of the 7th slide set was implemented to handle the ranking with a limited number of items. Here is a code snippet showing the implementation

```
def insert_posting(self, posting: Posting) -> None:
        if len(self._queue) < self._max_items:
                heapq.heappush(self._queue, (-posting.term_frequency, posting.doc_id, posting))
        elif -posting.term_frequency < self._queue[0][0]:
                heapq.heappop(self._queue)
                heapq.heappush(self._queue, (-posting.term_frequency, posting.doc_id, posting))
```

Note that the negative signs are necessary to turn the min heap into a max heap. For querying, the sorted postings list is retrieved using heapq's "nsmallest" method.

```
def get_postings(self) -> List[Posting]:
        return [element[2] for element in heapq.nsmallest(self._max_items, self._queue)]
```

3 Top 15 return functionality

4 Crawling

We used two libraries to handle web crawling and scraping. We opted for "Scrapy" as our web crawling framework and "Beatifulsoup" as our tool for scraping. Scrapy was a good choice for webscraping as it is usually used to scrapy a single domain, which is our case in this project.