Data Retrieval Project Report

Students:
Tzahi Kats 205888886 tzahikat@post.bgu.ac.il
Iftah Levy 209383306 Levyifta@post.bgu.ac.il

Git Repository: https://github.com/LevyIfta/DataRetrivelProject
Bucket: `ir_proj_205888886`

Body text index: `"postings_text/index.pkl`
Title index: “`postings_title/index.pkl"`
Anchor text index: `"postings_anchor_text/index.pkl"`

Key experiments:

Indexing experiments:
The first roadblock we had to experiment on was memory issue during indexing. Despite making sure to only load a single file at a time while indexing, we often ran out of memory mid-indexing causing the Kernel to restart (effectively wiping the indexing process). We had to experiment with different clusters settings and locations to hand-delete unnecessary arguments until we managed to find a working solution

Word models:
We checked a few different ML models in order to find words with similar meanings to ones in a given query.
First we checked PyDictionary. We checked its capabilities to give us synonyms, hyponyms and hypernyms. While its operation was smooth and easy, it showed us how complex word’s meanings can get. Many are the words that have many different meanings depending on connotation, so it was near impossible to reasonably identify the correct synonym. (for example the word “word” had 11 different meanings each with different synonyms {such as “intelligence”, “news”, “the Bible” })
Afterwards, we experimented with google’s Word2Vec. At first we attempted training a model based on a small corpus, but that gave us mixed results (the closest words to “water” were “soil” followed by “groundwater”).
Realising that perhaps we used too small of a corpus, we looked for pretrained models, and encountered Google’s pretraind Word2Vec model, which was trained on the entire

Wikipedia corpus (2015). While it definitely gave much better results, it would also be somewhat inconsistent. The model would suggest very similar words alongside weird phrases. “Sea” is very similar to “ocean” but no so much to “Creamsicle_orange”

Removal of rare words:
Quite soon we realized that there were a plethora of rare words, typos and numbers rarely in use. So we decided to try and find them and rebuild an index excluding them. The initial result were promising. On a small corpus, we found 387509 rare words out of 392438. 98.7% decrease in the number of words.
Even when tested on the entire corpus (so each word had many more opportunity to appear), we found that 97.5% of the words appeared <50 times.
However soon enough we understood that it wasn't as good as it seemed. While the common words were only 2.5% of all words, it also meant that they took 97.5% of all occurrences and so took a suitable size of the index. In addition to that, “rare words” might contain useful and popular words that are simply not used much in unrelated context. The word “turducken” is rarely used despite its wikipedia page being extremely popular.
In the end we decided that while removal of rare words would be beneficial, it does not justify recreating the entire index so we held off on implementing it.

Runtime:
Sadly, as we worked breath-wide with scrutiny, by the first time we ran a fully functionally search engine we already had satisfactory results (even on the very first query run times were <1 sec)

Good result:

Query: “Football”

```
[
  [
    "5708067",
    "GrassMaster"
  ],
  [
    "1662349",
    "Football (disambiguation)"
  ],
  [
    "23594875",
    "List of Football Associations by date of foundation"
  ],
  [
    "2493729",
    "Football association"
  ],
  [
    "23555338",
    "List of Australian rules football clubs in Western Australia"
  ],
  [
    "23555289",
    "List of Australian rules football clubs in South Australia"
  ],
  [
    "37491210",
    "North American football"
  ],
  [
    "33247878",
    "West Australian Country Football League"
  ],
  [
    "26187398",
    "Professional football"
  ]
]
```


Bad result:

Query: Tom yam soup

```
[
  [
    "49248790",
    "Nsala soup"
  ],
  [
    "56866",
    "Yam"
  ],
  [
    "47770304",
    "Tom yum"
  ],
  [
    "32104663",
    "Yam production in Nigeria"
  ],
  [
    "65811438",
    "Bing-yee Yam"
  ],
  [
    "23444539",
    "On Yam Estate"
  ],
  [
    "3152275",
    "Yam (vegetable)"
  ],
  [
    "45400645",
    "Asogli Te Za (Yam Festival)"
  ],
  [
    "53699466",
    "List of vegetable soups"
  ]
]
```


It seems the engine managed to easily find documents related to a single word query. Even on a multi word query it managed to return decent results per word.

On the other hand, it is very clear our engine struggle with multi word terms. In the case of “Tom yam soup” it struggled to understand that “tom yam” is a name of a soup. It gave multiple results related solely to “soup” or “yam”(a vegetable) but almost none related to the particular eastern dish.