Assignment 7

Information Retrieval and Web Search

Ahmed Rekik - 790063 & Tim Henning - 789242

January 23, 2018

1. Remerge or Rebuild?

A search engine's index of two year's news comments contains 1 million documents. Each day, about 5000 new comments are posted and about 100 comments are deleted (because of hate speech, insults, or defamation). The index is therefore updated with a Remerge strategy on a daily basis. However, once only, a large amount of outdated comments shall be deleted and the question is:

1.a. How many outdated comments would need to be deleted as part of the next daily update so that the Rebuild strategy becomes more efficient than Remerge?

In order to improve the running time of index rebuild such that the rebuild time becomes more efficient than the remerge time, we need to adapt the amount of out-dated indexed comments that need to be deleted. We know that:

$time_{rebuild} = c.d_{new} = c.(d_{old} - d_{delete} + d_{insert})$ for a constant c.

$time_{remerge} = c'.d_{new} = c'.(d_{insert} + \frac{1}{4}(d_{old} + d_{insert}))$ for a constant c'.

Let's suppose that constants here are negligible for our two running time functions, thus we take into consideration only our variables. We assume that $Time_{rebuild} = Time_{remerge}$ if:

$(d_{old} - d_{delete} + d_{insert}) = (d_{insert} + \frac{1}{4}(d_{old} + d_{insert}))$

In our case we want to get the minimum amount of outdated comments that need to be deleted in order to get a rebuild time time at least fast as the remerge running time, thus the following function holds:

$(d_{old} - d_{delete} + d_{insert}) \leq (d_{insert} + \frac{1}{4}(d_{old} + d_{insert}))$

by subtracting d_{insert} and $\frac{1}{4}d_{old}$ in the both functions we get:

$\frac{1}{4}d_{old} - d_{delete} \leq d_{insert}$, which leads us to the following inequation

$d_{delete} \geq \frac{3}{4}d_{old} - \frac{1}{4}d_{insert}$

We know that daily, we delete at least 100 undesired comments, let's suppose that $d_{outdated}$ is the number of outdated comments that need to be deleted to improve the running time of rebuild s.t $d_{delete} = d_{outdated} + 100$, therefore:

$d_{outdated} + 100 \geq \frac{3}{4}d_{old} - \frac{1}{4}d_{insert}$

We know that : $d_old = 1000000$ and the daily inserted comments $d_{insert} = 5000$, thereby:

$d_{outdated} \geq 750000 - 1250 - 100$

$d_{outdated} \geq 748650$

Since we want a **more** efficient rebuild running time, we have to delete **more** than 748 650 comments. $d_{outdated} > 748650$. $\Rightarrow$ 748651 outdated comments to delete

2. PageRank

2.a. Compute the pageRank score for the following network. Start by writing down the adjacency matrix. Assume a random jump probability of 0.15.

Let's define the following adjacency matrix of the given graph as follow:

$$\begin{array}{c|ccc} & A & B & C \\ \hline A & 0 & 1 & 0 \\ B & 1 & 0 & 0 \\ C & 0 & 1 & 0 \end{array}$$

We define the *pageRank* of a page u $PR(u)$ as follow:

$PR(u) = (1-p) + p.(\frac{PR(T_0)}{C(T_0)} + \frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$, where p is the probability ratio, $C(T_i)$ is the count of the outgoing edge from a page T_i and $T = \{T_0, T_1, ..., T_n\}$ is the set of pages that link to the page u.

In our case, all pages have only one outgoing edge, so $\forall u$ in our network $C(u) = 1$. To calculate the rankPage of our network, following equations hold:

1. $PR(A) = (1-p) + p.(PR(B))$
2. $PR(B) = (1-p) + p.(PR(A) + PR(C))$
3. $PR(C) = (1-p)$, since no vertex point to C, thereby $\mathbf{PR(C) = 0.85}$

For PR(B) we get after (1) and (3):

$PR(B) = (1-p)+p.((1-p)+p.(PR(B))+0.85) = (1-p)+p.(1-p)+p^2.PR(B)+p.0.85$

$(1-p^2).PR(B) = (1-p) + p.(1-p) + p.0.85 = 1.105 \Rightarrow PR(B) = \frac{1.105}{(1-p^2)}$

$\mathbf{PR(B) = 1.130}$

To finish, we get $\mathbf{PR(A) = 1.019}$.

2.b. PageRank is initialized with identical scores for each web page. What would be a better method to initialize pageRank so that less iterations are necessary?

PageRank is an algorithm that calculate the score of a page in terms of the score (in pageRank) of pages that link to that page. Thus, several iteration have to be iterated in order to compute the pageRank of each document and that task may be expensive at that rate knowing that in practice the amount of pageRank to compute is extremly high. Therefore reductions could be done in order to increase the computing time and to improve the ranking quality.

First reduction could be the removement of navigational links between pages. Most of pages nowaday include a navigational links such as menu that help the web surfer to navigate into the website. Those links could have an influence in the pageRank calculation and therefore increase strongly the pageRank variables into each others what deteriorates the efficency besides the fact that those links could manipulate the rank and decrease the ranking quality.

An other reduction could be the removement of nepostic links. Several people tend to share links of their pages in publishments out their pages (Forum, comments...). Those comments could be represented as superfluous not only for the pageRanking computation time but also harm the ranking. Their reduction would be a big boost for the PR computation eventhough it is hard (and expensive) to detect those links.

We could also reduce the kern of pages (documents) by considering that $pageRank(u) = pageRank(v)$, if u and v have the same in-comming edges and therefore they are similar. This reduction could save time in the pageRank computing by eliminating "duplicate" pageRank functions.

3. (Programming) Document Embeddings and Optimization

3.a. For at least one phrase query, one boolean query, and one keyword query, report the total processing runtime. In addition to that, report the name of the operation that you identified as the bottleneck (longest runtime).

Table 1: Query Performance

Query	runtime (ms)	bottleneck	runtime (ms)
'european union'	84.96	int_from_base64	68.0
party AND chancellor	162.31	int_from_base64	90.0
negotiate	17.05	int_from_base64	9.0

The bottleneck in our implementation is the method `int_from_base64(x)`. It is used to convert the IDs and position integer values, that were saved as base64, back to real integers. A more efficient binary encoding could solve this.

3.b. Use gensim to learn a vector representation for each comment. For one comment of your choice, find the most similar comment (not the exact same!) and print both.

Chosen comment:

```
  Pretty much all western production is in China.  Russia doesn't produce
anything except raw oil & gas.  Well, it does, but it doesn't have any impact
```

```
on the global market.  Russia is still accepting US $ for oil and gas while
China just loves to kiss US bottom.  If they really want to shake US hegemony,
than they should stop acting like US vassals.  The end of petrodollar will
not cause economic collapse of the USA, unless someone makes an larger impact
on their economy.
```

Most similiar comment found by gensim doc2vec:

```
  And the fact that it's everywhere and won't make the pranksters as much
profit.
```