Mikael Peltoketo 290513

Some notes: this is not perfect or 100% accurate implementation, but same architecture is achieved. This works with bit of abstraction. Main difference and my mistake is making only one word-file pair, so it's formed as word-file-file-.... But this ensures faster lookup time, and with encryption better confusion and privacy.

Experimental results

Experiment focused on analysing performance oof SID scheme described in the document. Implementation was done with Python 3 with Python Crypto -library and sqlite3. Implementation used SQL-database, so running the program doesn't rely on constant uptime. Test were run with same dataset as in the research paper.

Performance test were run on Windows 10 laptop with 16GB of RAM and AMD Ryzen 7 3700U -processor with clock speed of 2.3 GHZ, 4.0 GHZ boosted.

This is fearly realistic user computer case.

Dataset and it's realism

Dataset was same as in the research paper. Major difference was in keyword screening and parsing. All keywords shorter than 2 or longer than 50 were excluded, because they are either irrelevant or not real English words, further more to avoid SQL conflicts and lower the keyword count all ()[]{}/&%#"!?^*'¨><| and other were excluded. All keyword (and search words) were made lowercase. With this even thought unique keyword count went down, same amount of keywords were ultimately saved. This screening and parsing resulted in roughly 700 000-1 000 000 unique keywords in different sized datasets.

Dataset is realistic as is described in the paper, but I think it was too open with keyword selection. Normal user unlikely searches such random values as were allowed in database. So therefor dataset is ok, but it's use was misguided.

Indexing and Encryption

As expected this is most time consuming part, where time goes fairly evenly to database creation and file encryption.

Speed difference in paper and my implementation comes from newer Python and database versions and better keyword control.

Size	No of TXT Files	Unique Keywords	Time (min)
184 MB	425	666141	5.3
357 MB	815	753047	10.2
670 MB	1694	853957	25.2
1.0 GB	1883	878713	37.4
1.7 GB	2808	969261	66.7

In indexing 5 files had some errors (reading / encryption), so they aren't red, this error marginal was acceptable in this project in my opinion and could be fixed with different encoding.

Search

Search overall was extremely fast, on all dataset it was somewhere between 0-1 seconds. Testing this was done with 10 randomly chosen words from database.