On the Reproducibility of Experiments of Indexing Repetitive Document Collections

Antonio Fariña
antonio.farina@udc.es
Universidade da Coruña, CITIC
A Coruña, Spain

Miguel A. Martínez-Prieto
migumar2@infor.uva.es
University of Valladolid
Segovia, Spain

Francisco Claude
fclaude@recoded.cl
Universidad Diego Portales
Santiago, Chile

Gonzalo Navarro
gnavarro@dcc.uchile.cl
IMFD, DCC, University of Chile
Santiago, Chile

Juan J. Lastra-Díaz
jlastra@invi.uned.es
UNED
Madrid, Spain

Nicola Prezza
nicola.prezza@gmail.com
University of Pisa
Pisa, Italy

Diego Seco
dseco@udec.cl
IMFD, University of Concepción
Concepción, Chile

ABSTRACT

We summarize an already published work [3]. It consists in a companion paper that aims at allowing the exact replication of the methods, experiments, and results discussed in a previous work [2], where we proposed many techniques for compressing indexes which exploit that highly repetitive collections are formed mostly of documents that are near-copies. In this work, we show our replication framework (uiHRDC: available at https://github.com/migumar2/uiHRDC/), that permits to replicate the actual experimental setup from our parent paper with little effort. The experimentation was carefully explained, providing precise details about the parameters that can be tuned for each indexing solution. Finally, we also provided uiHRDC as a reproducibility package.

CCS CONCEPTS

• **Information systems** → *Data compression*; **Search index compression**.

KEYWORDS

Repetitive document collections, inverted index, self-index, reproducibility.

1 INTRODUCTION

Scientific advances have typically been disseminated in the form of scientific publications. Publications in Computer Science, usually present a new technique, algorithm, etc. that aims at improving upon the state of the art. Experimental results are typically provided as evidence of the achievements of the paper. Therefore, having public access to the original authors' setup (tools/source code, datasets, tuning parameters, etc.) used to drive such experimental evaluation becomes crucial to assess the validity of the results obtained and to allow other researchers to reuse this previous research as a baseline to compare with their future techniques.

According to the ACM [1], an experiment is *reproducible* only if an independent group of researchers can obtain the same results using artifacts that they developed independently. The ACM also defines *replicability* as a weaker type of reproducibility, where independent researchers must obtain the same results when using the artifacts provided by the original work's authors. Finally, *repeatability* refers to the possibility that the original author of a work could obtain the same results when reusing the original setup (same conditions, system,...); i.e. he can repeat the experiments and obtain the same results. The conclusion is that research works must be at least repeatable, and that achieving replicability would become desirable. Yet, that is not always simple as we must not only have source code, but also sometimes match the exact version of external dependencies, tune the parameters of the tools (such tuning is sometimes difficult to obtain from the original paper), or even keep the original computer without software/hardware updates. In our case, this was accomplished with a *Docker*-based solution.

2 OUR PROPOSAL

We focused on making the experiments of our previous work [2] replicable. In that parent paper, we tackled the problem of indexing repetitive document collections. We showed that the existing compressed posting lists representations commonly used for positional and non-positional inverted indexes are not well-suited to deal with highly repetitive collections. We provided new compressed posting lists representations (using run-length, lempel-ziv, and grammar based compression) that improved upon the state-of-the-art, and included also self-indexes in the comparisons. In our reproducible paper [3] we focused on:

- Briefly enumerating the 10 posting list representations included in the parent paper, and when applicable, we discussed the parameters of those techniques, and the actual values used to tune those parameters. Those techniques are: `rice`, `vbyte` (with three variants), `Simple9`, `PforDelta`, `QMX-coding`, `rice-runs`, `vbyte-Lzma`, `vbyte-Lzend`, `Repair` (with four variants). For all these techniques we used our

implementation of inverted indexes. We also included also the implementation of 4 techniques provided by a more recent work [4] (`partitioned-EliasFano`, `Opt-PForDelta`, `Interpolative-codes`, and `Varint-G8IU`).
- We briefly enumerated 6 self-indexing techniques used: `RLCSA`, `SLP`, `WCSA`, `WSLP`, `Lz77-index`, and `LzEnd-index`. We also showed how they were tuned in the original experiments.
- We described the original experimental framework, including both the datasets and the query patterns used, and finally describe how those experiments were run. We provided the actual source code for all the techniques compared and instructions regarding how to use them. This included the actual scripts that were used to create the different indexes at build time, and those scripts that leaded search experiments. Both datasets and source code were also made available.
- We clearly described the variables being measured. That is, the memory footprint of the different techniques (space requirements), and the CPU execution time at query time. Particularly when performing the operations *locate*, and *extract* with the available indexing/self-indexing techniques.
- We fixed some errors that were detected in the parent paper.
- We provided our reproducibility framework `uiHRDC` that permits to re-run all the experiments from the parent paper and to gather all the experimental results. It is discussed in Section 2.1.
- Finally, we used `uiHRDC` to automatically run the experiments in a new/different computer, and showed the results obtained. We discussed how the results permitted us to asses that the new results obtained actually draw the same conclusions as those in the parent paper.

2.1 uiHRDC Framework

The main contribution of this work was the careful creation of our `uiHRDC` *(universal indexes for Highly Repetitive Document Collections)* reproducibility framework. We included in it: (1) the source code and all the dependencies (with the actual versions) required to build the test programs; (2) the document datasets and query patterns used; (3) scripts to compile the source code for each technique; (4) scripts to build/create the compressed inverted indexes and self-indexes over the document collections. Such indexing/self-indexing structures are finally saved into disk; (5) scripts that load each representation from disk into memory and perform queries over such representation. Each run saves both space/time measurements into text result-files; (6) a script to compile all the source code, build all the indexes, and perform all the querying experiments (it includes repetitively launching steps 3 → 4 → 5 for each technique); (7) a script that collects all the result-files in first term. In same cases these are gnuplot-data files, in other cases they are simple text files containing the standard output of the search programs that we parse (using python scripts) to create well-formed gnuplot-data files. Then it creates all the figures from the parent paper using exactly the same styles to simplify the comparison between both versions; (8) an script to collect all those figures, and that using a latex template, generates a unique report in PDF format. In such report we also provide information regarding the machine in which the results were obtained (cpu, memory, o.s. version, etc) among other information. Figure 1 illustrates the workflow in `uiHRDC`.

Finally, we used *Docker* as the tool to provide a fully working environment where we tested our `uiHRDC` framework, hence guarantying replicability. We provided a Docker configuration file (*Dockerfile*) with instructions to create a container that exactly reproduces the configuration of the test framework and deploys uiHRDC. Basically, it installs ubuntu 14.04 (even though we also tested it successfully on ubuntu 16.04), with gcc 4.8, and installs from libraries such as libboost, to tools such as cmake, screen/byobu, textlive, gnuplot, or even an ssh-server to make ssh/sftp connections possible. We provide simple instructions to build our docker image, launch and connect to a docker instance (this is really simple as the user can simply connect by ssh as if the container were a running linux server), and then execute a script to deploy uiHRDC, run all the experiments, and get the final PDF report using a sftp connection.

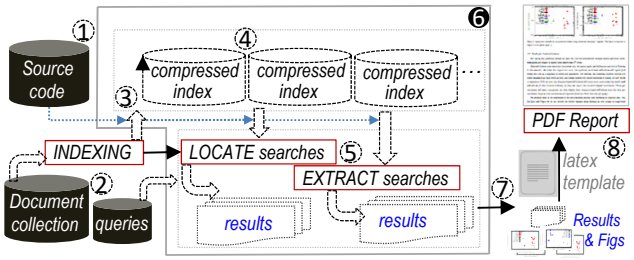

Figure 1: Workflow in `uiHRDC` to reproduce experiments.

3 CONCLUSIONS

We have briefly described all the techniques and their parameters from our original research [2]. We have also described the reproducibility framework `uiHRDC`, that includes the datasets, query sets, source code (and dependencies), as well as a set of scripts to automate the execution of all the experiments from [2]. They also automate the generation of a final PDF report where we collect the results (space/query-time) obtained by each technique and create all the figures from the parent paper. Finally, we provide a Docker container that reproduces our test enviroment to ensure that all our experiments can be replicated with little effort in any computer having Docker installed and matching our minimal RAM/disk requirements.

4 ACKNOWLEDGEMENTS

Supported by Xunta de Galicia/FEDER-UE [CSI: ED431G 2019/01 and GRC: ED431C 2017/58]; by Xunta de Galicia Conecta-Peme 2018 [Gema: IN852A 2018/14]; by MCIU-AEI/ FEDER-UE [Datos 4.0: TIN2016-78011-C4-1-R, BIZDEVOPS: RTI2018-098309-B-C32]

REFERENCES

[1] ACM. 2018. Artifact Review and Badging. https://www.acm.org/publications/policies/artifact-review-badging.
[2] Francisco Claude, A. Fariña, Miguel A. Martínez-Prieto, and Gonzalo Navarro. 2016. Universal Indexes for Highly Repetitive Document Collections. *Information Systems* 61 (2016), 1–23. https://doi.org/10.1016/j.is.2016.04.002
[3] Antonio Fariña, Miguel A. Martínez-Prieto, Francisco Claude, Gonzalo Navarro, Juan J. Lastra-Díaz, Nicola Prezza, and Diego Seco. 2019. On the reproducibility of experiments of indexing repetitive document collections. *Information Systems* 83 (2019), 181 – 194. https://doi.org/10.1016/j.is.2019.03.007
[4] Giuseppe Ottaviano and Rossano Venturini. 2014. Partitioned Elias-Fano Indexes. In *Proc. 37th Int. ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, NY, USA, 273–282. https://doi.org/10.1145/2600428.2609615