Reproducibility Report for ACM SIGMOD 2020 Paper: "Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks"

MIREK RIEDEWALD, Northeastern University, USA

The authors provided concise and sufficiently clear instructions where to find code and data, and how to repeat the main experiments from the paper. The structure of the code repository is generally self-explanatory, and together with the included README file, other researchers should be able to navigate it easily. Running the code mostly requires setting parameters in configuration files that control execution. The authors provided ready-to-run scripts with parameter settings that made it easy to reproduce the most important results from the original paper. However, due to an algorithm and code update after publication of the original paper, a few of the algorithm-quality-related numbers changed significantly. Trying to repeat experiments not covered by the provided scripts would require a reasonable additional effort to map settings discussed in the paper to the parameter values in the configuration files.

1 INTRODUCTION

We analyzed the reproducibility of the SIGMOD'20 paper titled "Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks" by Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan [1]. That paper proposes EmbDI, a technique to generate *local* embeddings of relational data. The authors demonstrate that the information captured by the embeddings improves the results of data integration algorithms such as schema matching and entity resolution. The main idea is to represent the database as a graph and to use random walks on this graph to generate "sentences" of table values (called *tokens*) that lead to an embedding that reflects the specific database content "near" a token. Experiments show improvements in F-measure over competing approaches, including pre-trained embeddings that are not database-specific.

To evaluate reproducibility, we followed the instructions provided by the authors, which stated the following: "To run the code and repeat the experiments, it is sufficient to call the `main.py` script as follows, replacing `*task*` with the appropriate experiment:

```
python3 main.py -d pipeline/config_files/*task*
```

This command will run all configuration files found in the provided path sequentially, printing out on screen the result obtained in each run." While these scripts were generally easy to run, they did not produce all results reported in the paper. However, arguably the most important results were covered. It would also be possible with reasonable effort to identify necessary modifications to the provided scripts so that other results from the paper can be reproduced.

Due to the probabilistic nature of the random walks, each execution of the experiments can create different embeddings. This in turn implies that results for data-integration tasks like schema matching and entity resolution will vary for each repeated execution. Our reproducibility effort generally confirmed the main message of the paper's experiments, but it also uncovered significant deviations in a few of the numbers related to algorithm quality. The authors confirmed that the latter is most likely caused by a code modification they added after publication of the original paper: *"Overall, the discrepancies between the results provided in the original paper and what has been tested with the code provided for reproducibility are likely direct consequences of a major modification made to the code, which modified how the algorithm handles cell values that can be split in multiple words. This led to a major change in the structure of the graph, which contains now a much larger number of nodes and edges. This change has led to better results in the [entity resolution (ER)] and [schema matching (SM)] tasks, and it is likely that the worsening of [embeddings quality (EQ)] results is a*

consequence of the change in how nodes are represented. Indeed, the EQ tests provided in the repository were designed with the earlier representation in mind. Since we consider the ER and SM results to be our major contribution, we decided that a loss in the EQ values would be an acceptable tradeoff."

2 SUBMISSION

The authors provided a concise PDF document with clear instructions for locating code and data. The code was easy to download from git repository https://gitlab.eurecom.fr/cappuzzo/embdi. The data and scripts for repeating the experiments can be downloaded from https://nextcloud.eurecom.fr/s/MwiRs8REfGRX5dm. Overall, the following was provided:

- A git repository at https://gitlab.eurecom.fr/cappuzzo/embdi with code, a detailed README, and test data.
- A PDF document `ReadMe_Embdi.pdf` with simplified instructions for the reproducibility evaluation. This document is not necessary for running the experiments, because all relevant information is also contained in the git repository, especially the README file.
- Datasets and scripts for repeating the main experiments from the paper at https://nextcloud.eurecom.fr/s/MwiRs8REfGRX5dm.

3 HARDWARE AND SOFTWARE ENVIRONMENT

All experiments in the original paper had been executed on a commodity laptop running Kubuntu OS. To remain close to that hardware, we used a laptop with similar specs. Since that laptop was running Windows 10, all reproducibility experiments were executed in an Ubuntu Linux virtual machine (VM) as summarized below.

Table 1. Hardware & Software environment

	Paper	Repro Review (VM in VMware Workstation 15 Player)
CPU	Intel i7-8550U	Intel i7-8650U
Cores	4	4 (1 available to VM)
GHz	1.8	1.9
RAM	32GB	16GB (8GB available to VM)
Storage	SSD	SSD
OS	Kubuntu 20.04	Ubuntu 20.04

4 REPRODUCIBILITY EVALUATION

4.1 Process

To evaluate the completeness of the author-provided instructions, we started with a bare-bones Ubuntu Linux 20.04 VM that had not been used for Python development and followed the instructions step by step. After installing all Python packages listed in file `requirements.txt` in the code repository, we executed all reproducibility scripts in directory `pipeline/config_files` in the order `embeddings_quality`, `entity_resolution`, and `schema_matching`. We did not run the embeddings-quality evaluation script for the Million Songs Dataset for several reasons: (1) it had not been included in the initial reproducibility material, (2) it is not used for the data integration evaluation in the original paper, and (3) the authors stated that due to its size, "training using the full set of sentences as in the paper would likely hit timeout (10h) ... on the test machine..."

After running the provided reproducibility scripts, we compared the observed numbers against those shown in the original paper. We discuss these results next.

4.2 Results

The `embeddings_quality` script produced results from Tables 2 and 6, but only for the authors' method (EmbDI) and not for competitors Basic, Node2Vec, and Harp. We were generally able to confirm the reported EmbDI quality numbers in Table 2. However, several numbers were off by more than 0.1, in some cases even more than 0.5, which is a large deviation given that the scale ranges from 0 to 1. Examples of large deviations include `MR` and `MC` for dataset `BB`, as well as all values for datasets `FZ` and `IM`. The authors confirmed that those differences are most likely caused by a modification of their code after publication of the original numbers (see above).

The running times reported in Table 6 for graph construction (column `G`) and generation of random walks (column `W`) were generally confirmed by our experiments, with our running times being slightly faster. Hence we initially found it surprising that our times for training of embeddings (column `E`) were several times slower than those reported in Table 6. The authors explained this phenomenon as follows: *"The most likely explanation for the difference in the training time of embeddings is the fact that the VM that was used for the reproducibility tests was running on one single CPU. The embeddings training algorithm is CPU-bound and strongly benefits from parallel processing, so the difference should disappear (or at least become smaller) by increasing the number of processors to be used. On the other hand, the graph preparation and random walks generation steps do not benefit from additional CPUs, which is the likely reason why the time didn't differ much between the two runs."* We were able to confirm this explanation by changing the number of cores for our VM to 4 and even 8 for some of the input datasets. This change indeed resulted in significant running-time reduction.

Script `schema_matching` produced results from Table 3, but again only for the authors' method EmbDI. In our experiment, the F-measure was 1.0 for each of the 8 datasets. This agrees with the numbers in the `EmbDI` column in Table 3, except for datasets `DS` and `IM`, where the table reports 0.5 and 0.78, respectively. This deviation is consistent with the authors' explanation regarding the code changes after publication of the original paper.

The author-provided script `entity_resolution` reproduces results from Table 5, but only for the row for $n_{top} = 10$. Our results generally confirm the numbers reported in that row in Table 5, with only a few numbers for datasets `AG` and `IA` deviating by more than 0.1 (again on a scale from 0 to 1). To reproduce the other rows from Table 5, one simply needs to change the value of parameter n_{top} in the script accordingly.

Using the provided scripts we generally were not able to reproduce results for the competing techniques, as well as the results shown in Table 4 and Figure 3. While this makes overall "coverage" of paper results by the author-provided reproducibility scripts lower than 25%, the most important results for EmbDI were confirmed.

5 SUMMARY

Anybody vaguely familiar with Python should be able to follow the instructions to run the code on a Linux machine, including VMs. Execution of the author-provided reproducibility scripts generally confirmed the main numbers reported for EmbDI. Since the provided scripts did not execute the competitors, the comparison between EmbDI and the state of the art was not reproducible.

REFERENCES

[1] Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In *Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). ACM, 1335–1349. https://doi.org/10.1145/3318464.3389742