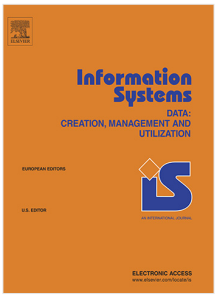
Journal Pre-proof



A reproducible POI recommendation framework: Works mapping and benchmark evaluation

Heitor Werneck, Nícollas Silva, Adriano Pereira, Matheus Carvalho, Alejandro Bellogín, Jorge Martinez-Gil, Fernando Mourão, Leonardo Rocha

PII: S0306-4379(22)00024-2
DOI: https://doi.org/10.1016/j.is.2022.102019
Reference: IS 102019

To appear in: *Information Systems*

Received date : 21 September 2021
Revised date : 8 March 2022
Accepted date : 10 March 2022

Please cite this article as: H. Werneck, N. Silva, A. Pereira et al., A reproducible POI recommendation framework: Works mapping and benchmark evaluation, *Information Systems* (2022), doi: https://doi.org/10.1016/j.is.2022.102019.

A Reproducible POI Recommendation Framework: Works Mapping and Benchmark Evaluation

Heitor Werneck[a], Nícollas Silva[b], Adriano Pereira*[b], Matheus Carvalho[a], Alejandro Bellogín[+c], Jorge Martinez-Gil[+d], Fernando Mourão[e], Leonardo Rocha[a]

[a] *Universidade Federal de São João del-Rei, São João del-Rei — Brazil*
werneck@aluno.ufsj.edu.br,{lcrocha,matheuscviana}@ufsj.edu.br
[b] *Universidade Federal de Minas Gerais, Belo Horizonte — Brazil*
{ncsilvaa,adrianoc}@dcc.ufmg.br
[c] *Universidad Autónoma de Madrid, Escuela Politécnica Superior, Madrid, Spain.*
alejandro.bellogin@uam.es
[d] *Software Competence Center Hagenberg, Softwarepark 32a, Hagenberg, Austria*
jorge.martinez-gil@scch.at
[e] *Seek, Belo Horizonte — Brazil*
fmourao@seek.com.au

Abstract

This work is a companion reproducibility paper that presents a framework to reproduce our previous experiments and results reported in Werneck et al. (2021). In that previous paper, we introduced a systematic mapping process of points-of-interest (POI) recommendation methods and provided a uniform evaluation methodology based on metrics covering different aspects besides accuracy. Due to the lack of reproducible and extensible benchmarks, our work introduces a reproducibility framework for POI methods based on a collection of Python software libraries and a Docker image. Our proposal is composed of: (1) a package to perform a protocol that reproduces our systematic mapping process (Werneck et al., 2021), containing all collected data, insightful views on current advances and opened challenges; and (2) an extensible benchmark to perform a protocol to reproduce experimental evaluations on POI recommendation, considering different datasets, metrics, and the strongest baselines in the literature. This work

*Corresponding author
Email address: adrianoc@dcc.ufmg.br (Adriano Pereira*)
+Reviewer

also demonstrates all processes required to instantiate its framework. Moreover, our work can be considered at least weakly reproducible, since we were able to reproduce the results of the previous paper, leading us to the same conclusions.

Keywords: POI Recommendation, Benchmark, Works Mapping

1. Introduction

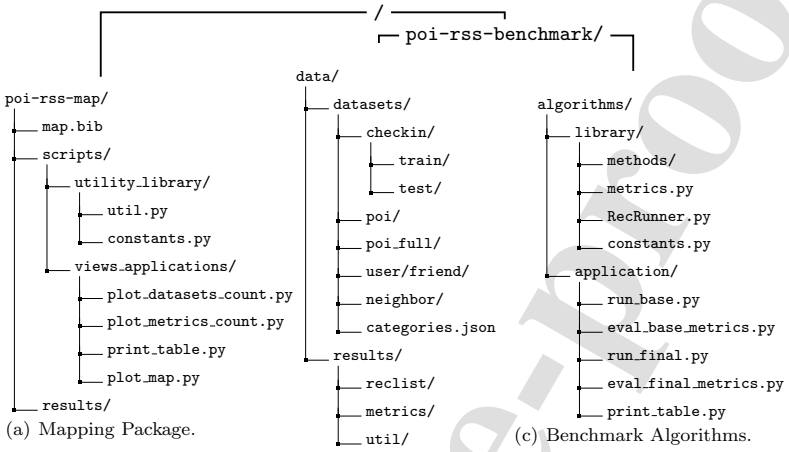
Nowadays, Location-Based Social Networks (LBSNs) have become an essential tool for users to share and discover new points-of-interest (POI), such as restaurants, museums, libraries, and others. There, users can express their current opinion about a place in distinct ways from traditional systems (Liu et al., 2019; Sun et al., 2020). For example, users can upload location-tagged photos to a social networking service, comment on an event at the exact place where the event is happening, share their present location on a website for organizing a group activity in the real world, record travel routes with GPS trajectories to share travel experiences in an online community, and many others. It has boosted the emergence of a large and growing number of distinct Recommender Systems (RSs) with characteristics and peculiarities for this scenario.

Our previous paper (Werneck et al., 2021) introduces a theoretical and experimental survey based on a systematic mapping of recent works published between 2017 and 2019 about POI recommendations. There, we highlighted: (1) Most works have focused on the generic POI recommendation problem, although we could identify a significant number of efforts addressing specific problems in the area, such as Next POI, and In/Out-of-Town Recommendations; (2) While the user preferences, geographical, temporal, and social information are the most explored in the current models, textual data is largely ignored; (3) Most of the new methods proposed by the selected papers apply Collaborative Filtering, Factorization, Probabilistic, and Hybrid strategies; (4) Most of the selected works have only evaluated the accuracy of the proposed methods; and (5) There is a lack of comparative studies among the existing proposals.

Regarding the last two highlights, we observed in our previous work that

despite the recognized importance of accuracy, there is a consensus in the RS community that other quality dimensions, such as novelty and diversity, are also essential to assess the practical effectiveness of recommendations. Moreover, most of the works are not concerned with evaluating their methods in distinct dimensions. These observations point out the potential damage to reproducibility and straightforward comparison of results in the area (Dacrema et al., 2021). In this sense, **this work aims to introduce a framework for a uniform and fair evaluation of POI recommendation methods to bridge the lack of reproducibility resources in this line of research.** Our framework is composed of a collection of Python software libraries and a Docker image in order to perform two protocols on POI recommendation area: **(1)** *Reproduction of the literature review*: It is performed by a package that reproduces the systematic mapping process (Werneck et al., 2021), with collected data, insightful views on current advances and opened challenges (Figure 1(a)). **(2)** *Reproduction of experimental evaluations*: It is composed of a package to support the evaluation of new POI recommendation methods with baselines from literature by using the same hardware/software to execute them; using distinct datasets with many check-ins and user's reviews; measuring metrics of accuracy, novelty and diversity; and applying statistical tests (Figure 1(b)).

For this paper, we use the framework to reproduce the same results reported in our previous paper (Werneck et al., 2021). However, it is possible to extend it by introducing new datasets, recommendation models, or evaluation metrics. Through our framework, researchers can evaluate their new methods of POI recommendation to get statistical results and compare them with previously existing methods under similar conditions. It also provides a way to make paper results reproducible since it configures a standard to perform experimental tests of POI recommendation methods, whose configuration details were previously defined and repeatedly tested. In an experimental test, we ran this reproducible package and achieved the same results, tables and plots by drawing the same conclusions as the previous paper on POI RSs (Werneck et al., 2021). Moreover, this framework is extensible to other methods and different



(a) Mapping Package. (b) Benchmark Data. (c) Benchmark Algorithms.

Figure 1: Framework directory structure.

analyses by easily accepting several new characteristics about them. It can also guide future proposals to address adequate, novel and relevant problems in the field that have not been tackled yet. Indeed, we ask future systematic mappings to increment, rework or extend our mapping with future advances in the area.

We organize the remainder of this paper as follows. Section 2 presents the protocol to reproduce the results of our systematic mapping process. Section 3 presents the protocol of our reproducibility benchmark. Finally, Section 4 presents the conclusions of this work.

2. Reproducible Protocol for the Literature Systematic Mapping

This section describes the first protocol of our reproducibility framework to allow the reproduction of our systematic mapping process, which is a scientific study methodology to summarize a research area in order to categorize and structure relevant efforts and results (Petersen et al., 2008). Unlike a systematic review, which focuses on in-depth descriptions and analysis of the literature, systematic mapping provides a coarse-grained overview. It brings undeniable benefits for reproducibility since each mapping step should be precise and deter-

ministically defined. Thus, it is possible to track research evolution by applying the same process at distinct moments over time.

Our systematic mapping process is consisted of five steps: (1) definition of the research questions to delimit its scope; (2) definition of study search parameters (i.e., language, period of publication, repositories, search engines, and keywords); (3) definition of inclusion and exclusion criteria of studies; (4) the classification scheme to organize the relevant studies; and (5) the paper mapping, in which we apply the classification scheme and a ranking function to the relevant studies. A more detailed explanation of each step can be found in our previous paper (Werneck et al., 2021).

The first stage aims to analyze a set of metadata manually collected from the papers of our Systematic Literature Review (SLR) (Werneck et al., 2021). Its package structure is centered in the bibliographic database extracted from the SLR (Figure 1 (a)). The code package comprises: (1) a bibliographic database, defined by using a *.bib* file; (2) a set of Python scripts to perform the visualization of our analyses; and (3) a directory to store the results. The *.bib* file works as a bibliography database, storing the metadata of selected papers in the SLR. Each entry of the *.bib* file records a paper by listing a metadata $m \in M$ and the value $v(m)$ measured by this characteristic. They are described in Table 1.

Metadata	**Description**
`problem`	Problem addressed by each work (e.g., Next-POI).
`methodology`	Methodology applied by the model (e.g., collaborative filtering, factorization).
`information`	Informations used to mitigate the problem (e.g., geographical, textual).
`dataset`	Datasets selected in the evaluation (e.g., Foursquare, Yelp).
`metrics`	Metrics used to evaluate the models (e.g., precision, intra-list diversity).
`model_name`	Refer to the corresponding work RSs proposals names.
`num_citations`	Number of citations extracted in Werneck et al. (2021).
`baselines`	Models used in the experimental evaluation.

Table 1: Description of metadata used to classify POI RSs works in the SLR.

In the *.bib* file parsing, we apply a specific Python library for parsing Bib-

TeX, named *bibtexparser*[2]. For the data visualizations we always use the *matplotlib*[3], a popular Python library. The most important files are (1) `util.py`, which contains a function to format the bibliographic database to allow further processing; (2) `constants.py`, which defines the datasets, parameters, metrics and also the language to produce results (available Portuguese and English); (3) `plot_datasets_count.py` to reproduce the plot around the frequency of each dataset; (4) `plot_metrics_count.py` to reproduce the plot around the frequency of each metric; (5) `print_table.py` to reproduce the mapping table; (6) and `plot_map.py` to plot the mapping in a bubble plot. Thus, every user can reproduce the results and figures found during the systematic literature review in the paper (Werneck et al., 2021). The `results` directory stores the resulting tables and figures obtained from the script applications execution.

Our framework is flexible and straightforward to other researches increment, rework and extend our mapping. We modularize this package into three components (i.e., bibliographic database, utility library and visualizations scripts). Future studies can use our literature mapping results to: expand it with new studies; increment the mapping with new classifications; and reuse visualizations in a new collection of works. Also, providing the raw data, it is more friendly to check the mapping to critics on the mapping results.

3. Reproducible Protocol for Benchmarks on POI Recommendation

We defined the framework using only one programming language – Python, to provide code consistency. We adopt the version *Python 3.8.10*. All the software is freely distributed for commercial and non-commercial usage under the Creative Commons License (CC BY 4.0). Aiming to ease the reproduction of experiments, we created a Docker image using Docker 20.10.7 with *Ubuntu 20.04.3*. This Docker image contains the framework and the datasets used to provide all the results of the previous paper (Werneck et al., 2021). The permanent image

[2] `https://github.com/sciunto-org/python-bibtexparser`
[3] `https://matplotlib.org/`

(i.e., the version used for this article to reproduce all results) is available at Mendeley Data. The files in it are presented in Table 2. The overview about all framework sources and the software information is summarized in Table 3.

Mendeley dataset data files	File size	Description
`benchmark-datasets.zip`	74 MB	Datasets used in the benchmark.
`benchmark-results.zip`	213 MB	Raw and processed benchmark output files.
`Dataset_Challenge_Dataset_Agreement.pdf`	99 KB	Yelp dataset terms of use.
`poi-rss.tar.gz`	2.5 GB	Docker image that can reproduce our experiments.

Table 2: Mendeley dataset content summary.

POI RSs Reproducible Framework	Description
GitHub benchmark repository	`https://github.com/heitor57/poi-rss.git`
Benchmark code version used in this work	benchmark
GitHub mapping repository & data	`https://github.com/heitor57/poi-rss-map.git`
Mapping code version used in this work	v1.1
Code language	Python 3.8.10
Code license	Creative Commons Attribution 4.0 International License (CC BY 4.0)
Docker version used to build the docker image	20.10.7
Docker image with all code & data used for this work	`https://data.mendeley.com/datasets/8sh5f96dfp/4`

Table 3: Framework overview about sources and software information.

The benchmark for POI recommendations structure is composed of 2 main components, `data` and `algorithms`, as shown in Figures 1 (b) and (c), including four modules: (1) the datasets, included in data component, providing raw material to the evaluation tests performed with our benchmark; (2) results, also included in data component, containing saved recommendation lists of the methods, metrics evaluations, result table, and general information; (3) library of recommender systems, in the algorithms component, containing metrics, methods, and execution managers; and (4) the end-user applications, in the algorithms component, containing command-line interfaces to execute the RS pipeline.

Regarding the datasets, we adopted the Yelp Open Dataset[4] for our benchmark, considering the US cities of Phoenix and Las Vegas filtered, as described in Werneck et al. (2021). The two cities have the most considerable portions of visit logs collected between 2004 and 2018 (details are shown in Table 4).

Dataset	# POIs	# users	# visit logs
Las Vegas	12,375	6,180	220,329
Phoenix	41,808	18,502	701,152

Table 4: Informations about the datasets of the benchmark obtained from the Yelp Open Dataset.

In the dataset file structure (`datasets`), the folders contain a file for each city (`{city}.pickle`) with their information to further simulate the recommendation scenario, as presented in Table 5.

Dataset artifact	Description
`checkin`	Contains two folders with the training and testing datasets.
`poi`	Includes the POI categories information without pre-processing.
`poi_full`	Holds the POI categories and their geographical locations (i.e., latitude and longitude), where categories are pre-processed with mapping to integer numbers $[0, +\infty]$.
`user/friend`	Contains files with user's friends.
`neighbor`	Has files with cities POI's neighbors POIs.
`categories.json`	Defines the datasets category tree.

Table 5: POI RSs benchmark datasets artifacts description.

The `results` (in data component) contains output files from algorithms executions. The output files of each recommendation task simulation are stored in `reclist` in a JSON file containing a list of JSON objects with the user identification, the ranking of POIs and their respective score, Table 6 shows the files generated by the execution of recommender systems. `metrics` folder is where the metric evaluation is saved as a JSON with the list of JSON objects containing the user identification and metrics value. Table 7 shows the files generated by evaluations of the recommendations lists. The results table is saved in the `util`

[4]Yelp Open Dataset — available at `https://www.yelp.com/dataset`.

Dataset	Algorithm	Recommendation list files generated by our benchmark
Las Vegas	USG	`lasvegas_usg_80_alpha_0_beta_0.2_eta_0.json`
Las Vegas	GeoSoCa	`lasvegas_geosoca_80_alpha_0.3.json`
Las Vegas	GeoMF	`lasvegas_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0.json`
Las Vegas	USG+Geo-Div	`lasvegas_usg_80_alpha_0_beta_0.2_eta_0_geodiv_20_div_weight_0.5.json`
Las Vegas	GeoSoCa+Geo-Div	`lasvegas_geosoca_80_alpha_0.3_geodiv_20_div_weight_0.5.json`
Las Vegas	GeoMF+Geo-Div	`lasvegas_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0_geodiv_20_div_weight_0.1.json`
Phoenix	USG	`phoenix_usg_80_alpha_0_beta_0.2_eta_0.json`
Phoenix	GeoSoCa	`phoenix_geosoca_80_alpha_0.3.json`
Phoenix	GeoMF	`phoenix_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0.json`
Phoenix	USG+Geo-Div	`phoenix_usg_80_alpha_0_beta_0.2_eta_0_geodiv_20_div_weight_0.5.json`
Phoenix	GeoSoCa+Geo-Div	`phoenix_geosoca_80_alpha_0.3_geodiv_20_div_weight_0.5.json`
Phoenix	GeoMF+Geo-Div	`phoenix_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0_geodiv_20_div_weight_0.1.json`

Table 6: Result files of running the algorithms on different datasets. These raw output files are stored in `/poi-rss-benchmark/data/results/reclist/` and processed later to generated the result tables.

folder (in .tex and .pdf), containing the datasets, metrics, methods, and the statistic test. Table 8 shows the post-processing output files of our benchmark.

We structure the `algorithms` directory as shown in Figure 1 (c), comprising: (1) a library containing methods, metrics, execution managers, and constants definition; and (2) the applications to execute simulations, evaluations, and visualization. In the library: the `methods` folder defines POI RS baselines (Table 9), `metrics.py` defines all the metrics (Table 10); `RecRunner` is a file with the *RecRunner* class that handles the RS simulation and evaluation pipeline; `constants.py` defines experiment settings and mapping of names of metrics, methods and datasets to use on visualizations.

In the application, we define command-line interfaces to the execution han-

Dataset	Algorithm	Output metric files of recommendations lists with size of 5, 10 and 20 generated by our benchmark
Las Vegas	USG	`lasvegas_usg_80_alpha_0_beta_0.2_eta_0_{5,10,20}.json`
Las Vegas	GeoSoCa	`lasvegas_geosoca_80_alpha_0.3_{5,10,20}.json`
Las Vegas	GeoMF	`lasvegas_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0_{5,10,20}.json`
Las Vegas	USG+Geo-Div	`lasvegas_usg_80_alpha_0_beta_0.2_eta_0_geodiv_20_div_weight_0.5_{5,10,20}.json`
Las Vegas	GeoSoCa+Geo-Div	`lasvegas_geosoca_80_alpha_0.3_geodiv_20_div_weight_0.5_{5,10,20}.json`
Las Vegas	GeoMF+Geo-Div	`lasvegas_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0_geodiv_20_div_weight_0.1_{5,10,20}.json`
Phoenix	USG	`phoenix_usg_80_alpha_0_beta_0.2_eta_0_{5,10,20}.json`
Phoenix	GeoSoCa	`phoenix_geosoca_80_alpha_0.3_{5,10,20}.json`
Phoenix	GeoMF	`phoenix_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0_{5,10,20}.json`
Phoenix	USG+Geo-Div	`phoenix_usg_80_alpha_0_beta_0.2_eta_0_geodiv_20_div_weight_0.5_{5,10,20}.json`
Phoenix	GeoSoCa+Geo-Div	`phoenix_geosoca_80_alpha_0.3_geodiv_20_div_weight_0.5_{5,10,20}.json`
Phoenix	GeoMF+Geo-Div	`phoenix_geomf_80_K_100_delta_50_gamma_0.01_epsilon_10_lambda__10_max_iters_7_grid_distance_3.0_geodiv_20_div_weight_0.1_{5,10,20}.json`

Table 7: Result of applying evaluation metrics for each algorithm in different datasets. Each row corresponds to 3 files, where the string "`{5,10,20}`" from the path name in each of these 3 files corresponds to 5, 10, and 20 (the sizes of the recommendation list). These raw output files are stored in `/poi-rss-benchmark/data/results/metrics/` and processed later to generated the result tables.

Post-processing output files	In previous paper (Werneck et al., 2021)
`benchmark_table.pdf`	table 6 and table 7
`benchmark_table.tex`	table 6 and table 7 (source file)

Table 8: Processed output files from the benchmark experiment, these files are stored at `/poi-rss-benchmark/data/results/util/` in the Docker image.

dler (i.e., `RecRunner`) to execute the RS evaluation pipeline. We define scripts to run traditional RS (`run_base.py`) and evaluate them (`eval_base_metrics.py`), given datasets and methods. Moreover, we define scripts to run post-processing RS (`run_final.py`) and evaluate them (`eval_final_metrics.py`), given datasets, traditional RS and post-processing RS. Finally, `print_table` prints a table with

the methods, metrics, and cities results with the Wilcoxon statistic test.

Baseline	Article	Description
USG	Ye et al. (2011)	USG is a unified collaborative filtering algorithm based on three main factors, namely: user preference (U); social influence from friends (S); and geographical influence from POIs (G).
GeoSoCa	Zhang & Chow (2015)	Explores the geographical, social, and categorical correlations among users and POIs to recommend POIs.
GeoMF	Lian et al. (2014)	Is a method that combines a matrix factorization and a geographical model based on user's activity areas and POIs influence areas.
Geo-Div	Han & Yamana (2017)	Selects active areas to filter POIs deemed as promising candidates to the diversification process by clustering overlapped visitable areas of each user.

Table 9: Information about the baselines of the benchmark.

Metric	Acronym	Work	Description
Precision	Prec	Ricci et al. (2011)	It is the percentage of relevant items recommended considering the number of recommended items.
Recall	Rec	Ricci et al. (2011)	It is the percentage of relevant items recommended considering the entire set of relevant items.
Coverage	Cov	Puthiya Parambath et al. (2016)	It assesses the diversity by the number of categories of the relevant items recommended for each user.
Intra-list distance	ILD	Vargas & Castells (2011)	It assesses the diversity by the dissimilarity between the item categories in the recommendation list.
Geographical proportional representation	PRg	Han & Yamana (2017)	It assesses the diversity from the geographic point of view, evaluating the distribution of recommended POIs in each subarea of the space.
Expected popularity complement	EPC	Vargas & Castells (2011)	It is measured by the expected number of relevant items not previously seen by the user (novelty).

Table 10: Information about the metrics of the benchmark.

4. Deploying the Experimental Setup & Replicating Results

As previously described, we created a Docker image that packs all code and data to reproduce the previous results from Werneck et al. (2021). Thus, this section presents all steps required to deploy this Docker image and reproduce all results.

Testing platform	Type	Operating Sys.	Configuration	Tested by
MINIMAL	Desktop	Ubuntu 20.04.3	1 Intel Core i3 (fourth generation or newer) or equivalent, 32 GB RAM, 100 GB mechanical disk	Authors
Platform1	Server	Ubuntu 20.04.3	1 Intel Core i7-3930K CPU @3.8 GHz, 32 GB RAM, 1 TB mechanical disk	Authors
Platform2	Desktop	Windows 10 Pro	1 Intel Core i7-8700 CPU @3.2 GHz, 32 GB RAM, 512 GB mechanical disk	Reviewer
Platform3	Desktop	Ubuntu 18.04.2	1 Processor with 16 cores, 128 GB RAM, 1.5 TB mechanical disk	Reviewer

Table 11: Testing platform used to reproduce our experiments.

First of all, to set up the environment of our reproducible framework is required to have Docker installed in a version equal or higher than that described in Table 3. This Docker image will require at least 6.46 GB and it will provide all code, input data and required software. In a Linux machine with bash and basic shell commands, it is possible to deploy this environment by the instructions described in Table 12.

We executed the experiment on only one computer, described in Table 11, and the runtime is shown in the Table 13.

4.1. Reproducing Results from the Systematic Literature Review

This part of our framework is the most lightweight and no particular machine is required. All codes are executed almost instantly, with low memory usage, and only 3 MB of disk space is sufficient to store the results. The workflow for reproducing the results of the systematic literature review is shown in Figure 2.

Step	Step-by-step instructions to deploy the Docker container

1	Install Docker in Ubuntu
	`$ sudo apt-get update`
	`$ sudo apt-get install apt-transport-https\`
	`ca-certificates curl gnupg lsb-release`
	`$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg \`
	`| sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg`
	`$ sudo add-apt-repository \`
	`"deb [arch=amd64] https://download.docker.com/linux/ubuntu\`
	`$(lsb release -cs) stable"`
	`$ sudo apt-get update`
	`$ sudo apt-get install docker-ce`
	`$ sudo apt-get install docker-ce-cli containerd.io`
	Verify if the Docker Engine is installed correctly
	`$ sudo docker run hello-world`

2	Open a terminal on Linux with bash, create and enter a folder to store the framework. `$ cd /home/ && mkdir poi-rss && cd poi-rss`

3	Download the *tar.gz* file containing the docker image to execute experiments from Mendeley Data.
	`$ wget "https://data.mendeley.com/public\`
	`-files/datasets/8sh5f96dfp/files/\`
	`8a4afc9d-5681-4a68-a30d-072f5790348f/file_downloaded"`
	`$ mv file_downloaded poi-rss.tar.gz`

4	Load the docker file and create a new container to posterior running of experiments.
	`$ docker load -i poi-rss.tar.gz`
	`$ docker run -itd --name poi-rss poi-rss bash`

Table 12: Instructions to deploy the Docker container.

Run	Testing platform	Running time	Tested by
1	Platform1	2160 min ≈ 1.5 days	Authors
2	Platform2	1265 min ≈ 21.08 hours	Reviewer
3	Platform3	703 min ≈ 11.72 hours	Reviewer

Table 13: Running time obtained with the testing platform.

Follow the step-by-step guide at Table 14 to reproduce the SLR results. The main output files are stored as illustrated in Figure 3.

4.2. Reproducing Results of POI Benchmark

In turn, the benchmark of POI recommendation has a moderated size and requires at least 1.2 GB of disk space to be stored. These experiments take around 36 hours to execute everything for the computer settings previously mentioned.

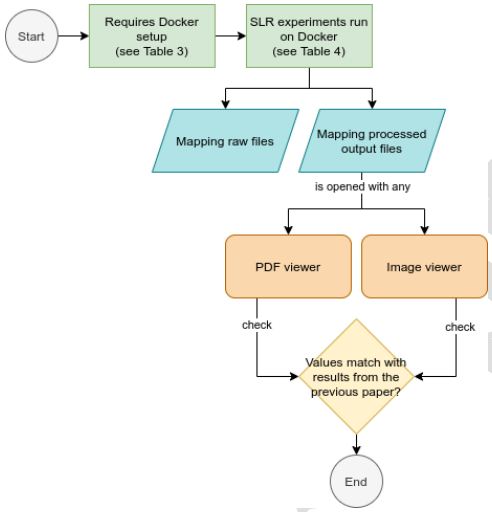


Figure 2: Reproducibility workflow using Docker in Ubuntu to reproduce SLR results of the paper (Werneck et al., 2021). The output files of this workflow are the same as the table and figures from the previous article (Werneck et al., 2021).

Step	Step-by-step instructions to reproduce the SLR experiments
1	Requires the docker setup described previously in Table 12.
2	Attach bash to the running container.
	`$ docker exec -it poi-rss bash`
3	Go to the project folder.
	`$ cd /poi-rss-map/scripts`
4	Execute python file to generate the systematic mapping results.
	`$ ./run_all`
5	Leave session of the Docker container to the host user session.
	`$ exit`
6	Copy result files from the container to the host.
	`$ docker cp poi-rss:/poi-rss-map/results slr-results`
7	Use any pdf viewer and image viewer to check results.

Table 14: Instructions to reproduce the SLR experiments.

It is expected that the experiment requires around 32 GB to run. To demonstrate the adaptability of our framework, we also include more methods and datasets than the original ones in the benchmark. They can be easily used with the command-line application. Figure 4 shows the workflow for reproducing

```
/poi-rss-map/data/results/
├── datasets_count.{png,eps}
├── metrics_count.{png,eps}
├── map.{png,eps}
└── map_table_full.{tex,pdf}
```

Figure 3: Main output files of the mapping results reproduction.

the benchmark results. The instructions to reproduce the results previously reported in Werneck et al. (2021) are described at Table 15.

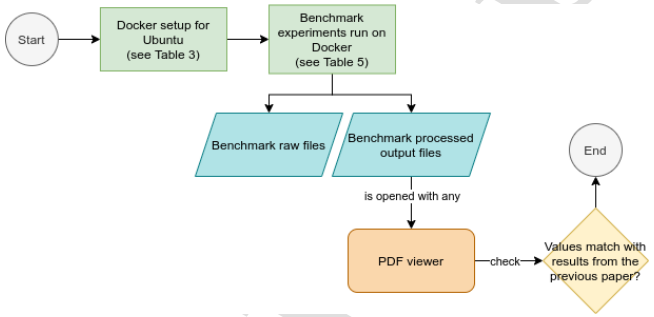


Figure 4: Reproducibility workflow using Docker in Ubuntu to reproduce the benchmark results of the paper (Werneck et al., 2021). The output files of this workflow are the same as the tables from the previous article (Werneck et al., 2021).

Executing our benchmark as presented, we can reproduce the results from the previous paper (Werneck et al., 2021), with the same conclusions. Moreover, our benchmark offers all the necessary structure to extend it, including new datasets, metrics and RS methods, with a complete experimental setup that ensures the quality and correct evaluation of new POI methods.

5. Extending and reusing our reproducible benchmark

Currently, to extend our benchmark to add new metrics, methods, and datasets, researchers should only implement the desirable code in the experimental controller (i.e., *RecRunner* class) and add the parameters/names in the `constants.py` file to handle the new feature in the benchmark. The code of

Step	Step-by-step instructions to reproduce the Benchmark experiments
1	Requires the docker setup described previously at Table 12.
2	Attach bash to the running container. `$ docker exec -it poi-rss bash`
3	Open the project folder. `$ cd /poi-rss-benchmark/algorithms/application`
4	Run the scripts to obtain the benchmark results: recommendation lists; metrics; and results table. The main parameters (i.e., cities, traditional recommenders and post-processing steps) are set specifically to reproduce the previous paper. `$ ./run_all`
5	Leave session of the Docker container to the host user session. `$ exit`
6	Copy result file from the container to the host. `$ docker cp poi-rss:/poi-rss-benchmark/data/results/util/benchmark_table.pdf .`
7	Use any pdf viewer to check results.

Table 15: Instructions to reproduce the SLR experiments.

the metric/method/dataset should be implemented in the correct module by following the pattern described in Section 3. Table 16 illustrates an example of adding a new recommendation system to the benchmark.

Step	Step-by-step instructions to implement a new recommender system
1	Create the configurations on the RecRunner class to manage the inputs.
2	In the RecRunner class, in the static method get _base_parameters, the recommendation model must expose its default parameters by a dictionary.
3	Set and create a handler to execute the recommendation model in RecRunner BASE_RECOMMENDERS attribute.
4	Set in CITIES_BEST_PARAMETERS dictionary at constants.py the parameters of the method to use in each city (dataset).
5	Set in RECS_PRETTY a name of the RS to appear in visualizations.

Table 16: Step-by-step guide to implement a RS in the benchmark framework.

6. New results generated by our experiments

This section aims to present the benchmark results, which are obtained following the workflow shown in Figure 4. This workflow was executed using the reviewer's test platforms described in Table 11, and the execution times are presented in Table 13. This section is concerned with the benchmark results,

considering the execution of the SLR workflow presented in Figure 2, to reproduce the SLR results. Both reviewers obtained identical results to the primary work (Werneck et al., 2021).

Table 17 presents the results of the benchmark that aims to reproduce Table 6 of the primary paper (Werneck et al., 2021), in which the results of the primary paper are presented (i.e., column "paper") together with the results obtained by the reviewers (i.e., column "rev1" and column "rev2"). In the same way, we present in Table 18 a comparison of the results achieved in the primary paper (Werneck et al., 2021) with the ones achieved by reviewers, regarding to Table 7 of the primary paper.

Finally, to guide the discussion about the degree of reproducibility of the framework in the next section, we present correlation measures of Pearson (r) and Spearman (ρ) in Table 19 between the paper and reviewers values of each metric presented in the Table 17. The value of the Pearson correlation evaluates the linear relationship of metrics of the primary paper (Werneck et al., 2021) with the metrics of the reviewers, while the Spearman correlation assesses the relationship between the ranking of metrics.

Las Vegas

	Prec@5			Rec@5			Cov@5			ILD@5			PRg@5			EPC@5		
Algorithm	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0.0333	0.0335	0.0336	0.0175	0.0175	0.0176	0.2260	0.2261	0.2261	0.5122	0.5123	0.5123	0.1787	0.1772	0.1770	0.2954	0.2952	0.2954
USG+GeoDiv	0.0370	0.0371	0.0370	0.0194	0.0195	0.0194	0.2200	0.2200	0.2200	0.5211	**0.5211**	**0.5212**	**0.4434**	**0.4427**	**0.4427**	0.4858	0.4844	0.4843
GeoSoCa	0.0298	0.0298	0.0298	0.0143	0.0143	0.0143	**0.2597**	**0.2597**	**0.2597**	0.4344	0.4344	0.4344	0.2099	0.2099	0.2099	0.7119	0.7119	0.7119
GeoSoca+GeoDiv	0.0275	0.0284	0.0284	0.0137	0.0138	0.0138	0.2560	**0.2569**	**0.2569**	0.4792	0.4771	0.4771	0.3326	0.3340	0.3340	**0.7209**	**0.7243**	**0.7243**
GeoMF	**0.0409**	**0.0393**	**0.0376**	0.0214	**0.0208**	**0.0199**	0.2303	0.2324	0.2345	0.5146	0.5166	0.5173	0.2412	0.2442	0.2434	0.5759	0.5745	0.5847
GeoMF+GeoDiv	**0.0411**	**0.0407**	**0.0396**	**0.0221**	**0.0217**	**0.0212**	0.2297	0.2305	0.2319	**0.5233**	**0.5252**	**0.5246**	0.3970	0.3984	0.3972	0.6147	0.6157	0.6247
	Prec@10			Rec@10			Cov@10			ILD@10			PRg@10			EPC@10		
Algorithm	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0.0309	0.0308	0.0308	0.0312	0.0311	0.0312	0.3424	0.3430	0.3430	0.5057	0.5059	0.5060	0.2406	0.2387	0.2385	0.3497	0.3482	0.3482
USG+GeoDiv	0.0331	0.0330	0.0329	0.0343	0.0341	0.0340	0.3289	0.3284	0.3286	0.5288	0.5289	0.5289	0.5525	0.5513	0.5512	0.5193	0.5176	0.5172
GeoSoCa	0.0253	0.0253	0.0253	0.0241	0.0241	0.0241	0.3511	**0.3511**	**0.3511**	0.4676	0.4676	0.4676	0.2504	0.2504	0.2504	0.7237	0.7237	0.7237
GeoSoCa+GeoDiv	0.0230	0.0239	0.0239	0.0225	0.0226	0.0226	**0.3537**	**0.3549**	**0.3549**	0.5074	0.5053	0.5053	0.4012	0.4026	0.4026	**0.7277**	**0.7304**	**0.7304**
GeoMF	**0.036**	**0.0363**	**0.0354**	**0.0377**	**0.038**	**0.0377**	0.3412	0.3382	0.3414	0.5323	0.5334	0.5340	0.3190	0.3214	0.3215	0.6002	0.6041	0.6090
GeoMF+GeoDiv	0.0353	**0.0354**	**0.034**	**0.0373**	**0.0377**	**0.0364**	0.3369	0.3376	0.3379	**0.549**	**0.5495**	**0.548**	**0.5596**	**0.5607**	**0.5607**	0.6443	0.6473	0.6556
	Prec@20			Rec@20			Cov@20			ILD@20			PRg@20			EPC@20		
Algorithm	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0.0280	0.0278	0.0278	0.0563	0.0559	0.0559	**0.4589**	**0.4587**	**0.4584**	0.5131	0.5133	0.5134	0.3036	0.3021	0.3017	0.4096	0.4080	0.4080
USG+GeoDiv	0.0289	0.0290	0.0289	0.0591	0.0592	0.0591	0.4453	0.4455	0.4454	0.5284	0.5284	0.5283	0.5560	0.5543	0.5541	0.5299	0.5289	0.5283
GeoSoCa	0.0209	0.0209	0.0209	0.0390	0.0390	0.0390	0.4509	0.4509	0.4509	0.5086	0.5086	0.5086	0.2916	0.2916	0.2916	**0.7344**	0.7344	0.7344
GeoSoCa+GeoDiv	0.0197	0.0205	0.0205	0.0373	0.0375	0.0375	0.4563	**0.4576**	**0.4576**	0.5243	0.5223	0.5223	0.4134	0.4143	0.4143	**0.7339**	**0.7362**	**0.7362**
GeoMF	**0.0313**	**0.0316**	**0.0311**	**0.0649**	**0.0655**	**0.0648**	0.4528	0.4531	0.4542	0.5513	0.5511	0.5517	0.3953	0.3961	0.3965	0.6353	0.6367	0.6418
GeoMF+GeoDiv	0.0293	0.0294	0.0290	0.0609	0.0615	0.0607	0.4510	0.4543	0.4548	**0.5674**	**0.5673**	**0.5668**	**0.6337**	**0.6348**	**0.6352**	0.6720	0.6726	0.6793

Phoenix

	Prec@5			Rec@5			Cov@5			ILD@5			PRg@5			EPC@5		
Algorithm	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0.0333	0.0236	0.0236	0.0175	0.0103	0.0103	**0.226**	**0.24**	**0.2399**	0.5122	0.3984	0.3984	0.1787	0.0744	0.0744	0.2954	0.4856	0.4856
USG+GeoDiv	**0.037**	0.0244	0.0244	**0.0194**	0.0105	0.0105	0.2200	0.2314	0.2314	**0.5211**	0.3857	0.3858	**0.4434**	**0.3348**	**0.3348**	0.4858	0.7013	0.7017
GeoSoCa	0.0167	0.0167	0.0167	0.0060	0.0060	0.0060	0.2123	0.2123	0.2123	0.4653	0.4653	0.4653	0.0978	0.0978	0.0978	0.8307	0.8307	0.8307
GeoSoca+GeoDiv	0.0150	0.0150	0.0150	0.0055	0.0055	0.0055	0.2037	0.2037	0.2037	0.5064	**0.5064**	**0.5064**	0.1915	0.1915	0.1915	**0.856**	**0.856**	**0.856**
GeoMF	0.0300	**0.0287**	**0.0291**	0.0124	**0.012**	**0.0121**	0.2176	0.2171	0.2191	0.4602	0.4610	0.4607	0.1427	0.1440	0.1440	0.7023	0.7032	0.7042
GeoMF+GeoDiv	0.0287	**0.028**	**0.0285**	0.0119	**0.0119**	**0.0118**	0.2182	0.2172	0.2190	0.4718	0.4723	0.4732	0.3010	0.3019	0.3020	0.7584	0.7574	0.7577
	Prec@10			Rec@10			Cov@10			ILD@10			PRg@10			EPC@10		
Algorithm	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0.0203	0.0202	0.0202	0.0177	0.0177	0.0177	0.3147	0.3148	0.3148	0.3908	0.3909	0.3909	0.1144	0.1137	0.1136	0.5201	0.5199	0.5200
USG+GeoDiv	0.0215	0.0215	0.0215	0.0187	0.0186	0.0186	**0.3212**	**0.3212**	**0.3212**	0.3977	0.3972	0.3971	0.4009	0.3998	0.3998	0.7055	0.7043	0.7044
GeoSoCa	0.0167	0.0145	0.0145	0.0060	0.0101	0.0101	0.2123	0.2839	0.2839	0.4653	0.4959	0.4959	0.0978	0.1258	0.1258	0.8307	0.8365	0.8365
GeoSoCa+GeoDiv	0.0150	0.0136	0.0136	0.0055	0.0096	0.0096	0.2037	0.2804	0.2804	**0.5064**	**0.5263**	**0.5263**	0.1915	0.2283	0.2283	**0.856**	**0.8561**	**0.8561**
GeoMF	**0.026**	**0.0255**	**0.0257**	**0.0214**	**0.021**	**0.0211**	0.3038	0.3042	0.3050	0.4796	0.4801	0.4804	0.2002	0.2014	0.2012	0.7258	0.7292	0.7295
GeoMF+GeoDiv	**0.0245**	**0.0245**	0.0244	0.0201	**0.0205**	0.0199	0.3061	0.3054	0.3072	0.5030	0.5035	0.5041	**0.4181**	**0.4193**	**0.419**	0.7808	0.7810	0.7810
	Prec@20			Rec@20			Cov@20			ILD@20			PRg@20			EPC@20		
Algorithm	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0.0176	0.0176	0.0176	0.0303	0.0303	0.0303	**0.4028**	**0.4029**	**0.4029**	0.3842	0.3841	0.3841	0.1702	0.1692	0.1691	0.5628	0.5627	0.5626
USG+GeoDiv	0.0192	0.0192	0.0192	0.0331	0.0330	0.0330	0.4002	**0.4002**	**0.4002**	0.4015	0.4010	0.4010	0.3899	0.3886	0.3885	0.6932	0.6924	0.6926
GeoSoCa	0.0125	0.0125	0.0125	0.0171	0.0171	0.0171	0.3800	0.3800	0.3800	0.5335	0.5335	0.5335	0.1555	0.1555	0.1555	0.8447	0.8447	0.8447
GeoSoCa+GeoDiv	0.0123	0.0123	0.0123	0.0168	0.0168	0.0168	0.3801	0.3801	0.3801	**0.5438**	**0.5438**	**0.5438**	0.2323	0.2323	0.2323	**0.8542**	**0.8542**	**0.8542**
GeoMF	**0.0225**	**0.0223**	**0.0225**	**0.0363**	**0.0363**	**0.0366**	0.3943	0.3948	0.3949	0.5053	0.5052	0.5057	0.2606	0.2617	0.2620	0.7541	0.7562	0.7568
GeoMF+GeoDiv	0.0216	**0.0215**	0.0217	**0.0358**	**0.0356**	**0.0359**	0.3981	0.3984	0.3995	0.5198	0.5198	0.5206	**0.4451**	**0.4459**	**0.4456**	0.7927	0.7927	0.7929

Table 17: Reproduction of the Table 6 of our primary paper (Werneck et al., 2021) contrasting all implemented baseline methods, using the Yelp dataset, and considering six distinct metrics on top-k (i.e., 5, 10, 20) recommendation lists. The best results with statistical significance when applying a Wilcoxon test with p-value = 0.05 are marked in **bold**. The column "paper" stands for the primary paper values, "rev1" and "rev2" are the results obtained by the reviewers.

Algorithm	Prec			Rec			Cov			ILD			PRg			EPC			Total		
	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2	paper	rev1	rev2
USG	0	0	0	0	0	0	3	3	3	0	0	0	0	0	0	0	0	0	**3**	**3**	**3**
USG+GeoDiv	1	0	0	1	0	0	1	2	2	1	1	1	2	2	2	0	0	0	**6**	**5**	**5**
GeoSoCa	0	0	0	0	0	0	1	2	2	0	0	0	0	0	0	1	0	0	**2**	**2**	**2**
GeoSoCa+GeoDiv	0	0	0	0	0	0	1	3	3	2	3	3	0	0	0	6	6	6	**9**	**12**	**12**
GeoMF	3	6	6	6	6	6	0	0	0	0	0	0	0	0	0	0	0	0	**9**	**12**	**12**
GeoMF+GeoDiv	2	5	3	3	5	4	0	0	0	3	3	3	4	4	4	0	0	0	**12**	**17**	**14**

Table 18: Reproduction of the Table 7 of our primary paper (Werneck et al., 2021) that contains the number of times each baseline was the top performer. The column "paper" stands for the primary paper values, "rev1" and "rev2" are the results obtained by the reviewers.

	Las Vegas				Phoenix			
	rev1		rev2		rev1		rev2	
Metric	r	ρ	r	ρ	r	ρ	r	ρ
Prec@5	0.9950	1.0000	0.9810	1.0000	0.7971	0.5429	0.7815	0.5429
Rec@5	0.9983	1.0000	0.9898	1.0000	0.7220	0.5429	0.7222	0.5429
Cov@5	0.9988	1.0000	0.9949	1.0000	0.9254	1.0000	0.9480	0.9429
ILD@5	0.9993	1.0000	0.9992	1.0000	-0.5382	-0.4286	-0.5382	-0.4286
PRg@5	0.9999	1.0000	0.9999	1.0000	0.9005	0.8286	0.9005	0.8286
EPC@5	0.9999	1.0000	0.9994	1.0000	0.9423	1.0000	0.9426	1.0000
Prec@10	0.9979	1.0000	0.9970	1.0000	0.9920	1.0000	0.9930	1.0000
Rec@10	0.9995	1.0000	0.9989	1.0000	0.9971	1.0000	0.9984	1.0000
Cov@10	0.9888	1.0000	0.9984	1.0000	0.9619	1.0000	0.9713	1.0000
ILD@10	0.9994	0.9429	0.9990	0.9429	0.9764	0.9429	0.9771	0.9429
PRg@10	0.9999	1.0000	0.9999	1.0000	0.9936	0.8857	0.9936	0.8857
EPC@10	0.9999	1.0000	0.9994	1.0000	0.9998	1.0000	0.9998	1.0000
Prec@20	0.9978	1.0000	0.9984	1.0000	0.9999	1.0000	1.0000	1.0000
Rec@20	0.9996	1.0000	0.9999	1.0000	1.0000	1.0000	0.9999	1.0000
Cov@20	0.9616	0.9429	0.9467	0.9429	0.9998	1.0000	0.9987	1.0000
ILD@20	0.9994	1.0000	0.9992	1.0000	1.0000	1.0000	1.0000	1.0000
PRg@20	1.0000	1.0000	0.9999	1.0000	1.0000	1.0000	1.0000	1.0000
EPC@20	1.0000	0.9429	0.9996	0.9429	1.0000	1.0000	0.9999	1.0000

Table 19: Computation of Pearson (r) and Spearman (ρ) correlation between the vectors defined by the values of each metric in Table 17. It is computed the correlation of reviewers results (rev1 and rev2) with the primary paper results (Werneck et al., 2021).

7. Discussion of the new experimental results

This section aims to discuss the differences between the new results obtained with the results from our primary paper (Werneck et al., 2021). The minor differences in the results were expected because the primary paper was not made in a reproducible environment, using a Docker image, for instance. However, the major conclusion made by our previous paper remains the same.

Among the results reported by the reviewers, we identified a small difference with the primary paper. In the primary paper, the USG algorithm showed higher values of coverage (Cov) than those currently reported (see Table 17 and Table 18). This difference is due to a recent change to one of the Python libraries used (i.e., NumPy) to calculate USG math operations. In a recent release, the accuracy of NumPy operations has increased from 64-bit to 128-bit. Thus, when the reviewers performed the experiments with the new version of the library, the items presented in the top-5 recommendation list for the Phoenix database became different from the primary article. Other small differences from the Table 17 and Table 18 are related to randomness sources of GeoMF and USG algorithms. More specifically, the geographical component of USG is non-deterministic, as it depends on random initialization of two variables, namely $w0$ and $w1$, which are used in a power-law function to fit the data, presented in Equation 8 of the article (Ye et al., 2011). On the other hand, GeoMF(Lian et al., 2014) initializes its P and Q matrices, which are users' latent vectors and POIs' latent vectors, respectively, with a probabilistic uniform distribution. Such changes harmed the correlation presented by the Spearman and Pearson indicators for the USG results in the ILD@5 metric in the Phoenix dataset. However, the behavior identified by the other algorithms for the evaluation metrics remained stable. Furthermore, after the authors performed a new execution with the new experimental environment, considered by the reviewers, the results achieved are identical to the results reported by the reviewers in the new Table 17. It endorses the high consistency in the reproductive character of the current article.

Considering the Definition 1 provided in the reproducibility guidelines, regarding the comparison with the primary paper, our computational experiment may be considered as weakly reproducible once: (1) the Spearman correlation between the original and reproduced results is equal to 1; and (2) the Pearson correlation value is high enough to allow the confirmation of all previously reported conclusions. We noticed that some metrics did not fit the previous definition of weakly reproducible: Prec@5, Rec@5, Cov@5, Cov@20, ILD@5, ILD@10, PRg@5 and PRg@10. As explained in the previous paragraph, the results identified so far were not fully reproducible due to the modifications related to the USG equations. However, the major conclusion highlighted in the primary paper remains the same. Indeed, if we contrast the results reported by the reviewers, we can notice that the results are almost equal. Specifically, the correlation between all values from both results (i.e., rev1 and rev2) has a Pearson and Spearman correlation equal to 1. Thus, all these new experiments are at least weakly reproducible.

Definition 1 (Weakly reproducible experiment)**.** *Given a set of previously reported experimental results and conclusions, a computational experiment is weakly reproducible if the Spearman rank correlation between the original and reproduced results is equal to 1, and their Pearson correlation value is high enough to allow the confirmation of all previously reported conclusions, even if the reproduced results do not reproduce all results exactly. Thus, weak reproducibility is a performance-rank-preserving notion.*

8. Conclusions

In this work, we presented and described a reproducible framework split into two packages: (1st) a mapper of the main works recently published in the POI recommendation literature; (2nd) a benchmark with different recommendation approaches, a usual dataset and all evaluation methodology. The first one also contains the bibliographic mapping data and visualizations scripts that can be further extended for other Systematic Literature Reviews. The second one pro-

vides a well-organized methodology to evaluate POI recommendation systems with baselines and metrics ready to use in current research. They are bundled in a Docker image that contains all required software used to reproduce results from our previous paper and achieve the same conclusions of Werneck et al. (2021). With a new execution of the benchmark to reproduce the results of the previous article, done by two reviewers, we report only small differences concerning the results presented in the main article (Werneck et al., 2021), leading us to consider our work at least weakly reproducible.

Moreover, this framework can also be applied to other researches for future explorations in the POI field. Researchers can easily add new baselines, metrics and datasets to the framework and compare them with the existing ones.

9. Revision comments

We would like to thank the authors for providing this valuable reproducibility framework as a first attempt to create a standard for experimentation in this domain. We think that the present framework presents a triple contribution: a) it allows the reproduction of the results obtained in the original paper, b) it allows comparison with the methods that form the state-of-the-art at present, c) it facilitates the development and evaluation of new methods in the context of POI recommendation. This is especially interesting because the challenge of POI recommendation is expected to attract much more research in the coming years. The large amount of content and platforms that currently exist need to offer their users mechanisms that facilitate the identification of items that may be of interest to them. Therefore, the development of methods and tools in this direction is highly desirable. The community will benefit greatly from reproducible research that allows past results to be used to build new solutions.

The present work confirms that the realization of reproducible science is far from being a trivial task. Even though reproducibility is one of the requirements present from the beginning, this is especially true in the field of computer science, where developments rely on a heterogeneous set of tools and technolo-

gies that are not static but evolve over time, creating an underlying problem of inter-version compatibility between libraries, systems, and components. Not to mention the cold starts based on the random information generation of a large majority of optimization or learning algorithms. In this specific scenario, we have encountered some of these recurring issues. In the following, we explain in detail our experience in evaluating this framework. Thus, we explain which design and implementation decisions seem to us successful and which factors have complicated our task in reproducing the experiments.

Among the design decisions that we believe to be sound and that facilitate the task of assessing reproducibility are the following:

- Docker containers make it much easier to reproduce the experiments since the working environment is known in advance, and both the libraries and the access paths are correctly configured.
- Python as a programming language includes easy syntax, an abundance of libraries, good online documentation and community support, and so on. Furthermore, being an interpreted language, it is possible to open the source file and check the operation mode when in doubt about a particular action.
- The generation of documentation, including tables, in a LaTeX format and the plotting of the results using figures in a standard format, facilitate both the interpretation of the results and their integration in other works or the elaboration of related material.

However, despite the authors followed these standards and had reproducibility as a goal, we detected some problems during our review:

- One of the most critical problems discussed with the authors was the confirmation that the dataset could be shared. This involved reading legal documents regarding the terms of use of the dataset to find the best conditions under the dataset and subsequent artifact (Docker instance) could be shared with the community.

- Even when using a Docker instance, some commands included in the instructions provided by the authors to run the experiments were not correct. For example, the name of the scripts or executables was a different one or they were located in a different path.

- It was not possible to check for strong or weak reproducibility constraints, since tables with results were not included. This was fixed in the revised version and actually allowed to compute correlations between original and reproduced results, providing a more complete analysis of the extent of the repeated experiments.

- Sometimes console reporting is difficult to interpret, as it requires in-depth knowledge of the algorithms being tested. Most of the doubts can be solved by consulting the source code. But a more detailed associated documentation is recommended for user level operators.

In conclusion, we fully support the choice of using containers for these kinds of reproducibility experiments. Using these technologies makes it extremely easy to reproduce previous experiments and obtain comparable results since several issues such as the availability of the correct version of the libraries or the required dependencies disappear. Based on this experience, it can be stated that the proposed framework agrees with the results reported by the authors in the original paper. In particular, the results differ slightly, so this would be the case of weak reproducibility. Such discrepancy has been identified and explained in the paper. We want to thank the authors for their considerable effort to provide a valuable software framework to the research community of this increasingly popular research domain, POI recommendation. We want to encourage the community to integrate their methods within this solution as a final note. The framework is fully prepared for its inclusion, and the automatic generation of statistics related to the results obtained should make it possible to compare them easily.

Acknowledgments

This work was partially supported by CNPq, CAPES, Fapemig, AWS and INWEB.

References

Dacrema, M. F., Boglio, S., Cremonesi, P., & Jannach, D. (2021). A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems (TOIS)*, *39*, 1–49. doi:10/ghwb82.

Han, J., & Yamana, H. (2017). Geographical diversification in poi recommendation: toward improved coverage on interested areas. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (pp. 224–228).

Lian, D., Zhao, C., Xie, X., Sun, G., Chen, E., & Rui, Y. (2014). GeoMF: Joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* KDD '14 (pp. 831–840). New York, NY, USA: Association for Computing Machinery. doi:10/gh7hmj.

Liu, C., Liu, J., Wang, J., Xu, S., Han, H., & Chen, Y. (2019). An attention-based spatiotemporal gated recurrent unit network for point-of-interest recommendation. *ISPRS International Journal of Geo-Information*, *8*, 355. doi:10/gmmq8j.

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering* EASE'08 (p. 68–77). Swindon, GBR: BCS Learning & Development Ltd.

Puthiya Parambath, S. A., Usunier, N., & Grandvalet, Y. (2016). A coverage-based approach to recommendation diversity on similarity graph. In *Proceedings of the 10th ACM Conference on Recommender Systems* (pp. 15–22).

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender systems handbook* (pp. 1–35). Springer.

Sun, K., Qian, T., Chen, T., Liang, Y., Nguyen, Q. V. H., & Yin, H. (2020). Where to go next: Modeling long-and short-term user preferences for point-of-interest recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 214–221). volume 34.

Vargas, S., & Castells, P. (2011). Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems* (pp. 109–116).

Werneck, H., Silva, N., Viana, M., Pereira, A. C., Mourão, F., & Rocha, L. (2021). Points of Interest recommendations: Methods, evaluation, and future directions. *Information Systems*, *101*, 101789. doi:10/gmmq79.

Ye, M., Yin, P., Lee, W.-C., & Lee, D.-L. (2011). Exploiting geographical influence for collaborative point-of-interest recommendation. SIGIR '11 (pp. 325–334). New York, NY, USA: Association for Computing Machinery. doi:10/dt5zgx.

Zhang, J.-D., & Chow, C.-Y. (2015). GeoSoCa: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. SIGIR '15 (pp. 443–452). New York, NY, USA: Association for Computing Machinery. doi:10/gmmq8b.

Article Highlights:

- A companion reproducibility paper about POI recommendation;
- An extensible benchmark about POI recommendation;
- A collection of Python software libraries and a Docker image;
- A package to perform a protocol that reproduces our systematic mapping process;
- A detailed reproducibility analysis of the primary work.

Conflict of Interest

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

None.