Reproducibility of the Neural Vector Space Model via Docker[*]

Nicola Ferro[0000−0001−9219−6239], Stefano Marchesin[0000−0003−0362−5893], Alberto Purpura[0000−0003−1701−7805], and Gianmaria Silvello[0000−0003−4970−4554]

Department of Information Engineering
University of Padua
Padua, Italy
{name.surname}@unipd.it

Abstract. In this work we describe how Docker images can be used to enhance the reproducibility of Neural IR models. We report our results reproducing the Vector Space Neural Model (NVSM) and we release a CPU-based and a GPU-based Docker image. Finally, we present some insights about reproducing Neural IR models.

1 Introduction

Reproducibility of models and systems is central for the verification of scientific results ans it is one of the cornerstones of the system of sciences. In the field of Information Retrieval, reproducibility has been the object of thorough analyses and efforts; only in the last two years we have seen the rise of many reproducibility-oriented events like the CENTRE evaluations at CLEF [2], NTCIR [7] and TREC [8], the SIGIR task force to implement ACM's policy on artifact review and badging and the *Open-Source IR Replicability Challenge at SIGIR 2019 (OSIRRC 2019).*

In this paper, we advocate OSIRRC 2019's vision that is to build Docker-based[1] infrastructures to replicate results on standard IR *ad hoc* test collections. Docker is a tool that allows for the creation and deployment of applications via images containing all the required dependencies. Relying on a Docker-based infrastructure to replicate the results of existing systems, helps researchers to avoid all the issues related to system requirements and dependencies. Indeed, *Information Retrieval (IR)* platforms such as Anserini, Terrier, or text matching libraries such as MatchZoo rely on a set of software tools, developed in `Java` or `Python` and based on numerous libraries for scientific computing, which have all to be available on the host machine in order for the applications to run smoothly.

We explore the use of Docker images for the reproducibility of *Neural IR (NeuIR)* models, which is a challenging domain that has seen only a few reproducibility efforts so far [1, 5]. NeuIR models are particularly hard to reproduce

[*] The full paper has been originally presented at the OSIRRC@SIGIR workshop [3]
[1] https://www.docker.com/.

because they are highly sensitive to parameters, hyper-parameters, and pre-processing choices. Also, these models are usually compatible only with specific versions of the libraries that they rely on (e.g., `Tensorflow`) because these frameworks are constantly updated. The use of Docker images is a possible solution to avoid these deployment issues on different machines as it already includes all the libraries required by the contained application.

For this reason, (i) we propose a Docker architecture that can be used as a framework to train, test, and evaluate NeuIR models; and, (ii) we show how this architecture can be employed to build a Docker image that replicates the *Neural Vector Space Model (NVSM)* [9], a state-of-the-art unsupervised neural model for *ad hoc* retrieval.

2 Background

Repeatability, replicability, and reproducibility are fundamental aspects of computational sciences, both in supporting desirable scientific methodology as well as sustaining empirical progress. These concepts have been discussed and analyzed in depth in [4], that focuses on the core issues and approaches to reproducibility in several fields of computer science. Relevant to these concepts is the *Platform, Research goal, Implementation, Method, Actor and Data (PRIMAD)* model, which tackles reproducibility from different angles. The PRIMAD paradigm has been adopted by the IR community, where it has been adapted to the context of IR evaluation – both system-oriented and user-oriented. In this context, our contribution lies between replicability and reproducibility. Indeed, we rely on the NVSM implementation available at [6] and described in [5] to replicate the results of a reproduced version of NVSM.

NVSM is a state-of-the-art unsupervised model for *ad hoc* retrieval. The model achieves competitive results against traditional lexical models and outperforms state-of-the-art unsupervised semantic retrieval models, like the Word2Vec-based models. NVSM jointly learns distinct word and document representations by optimizing an unsupervised loss function which minimizes the distance between sequences of n-grams and the documents containing them. Such optimization objective imposes that n-grams extracted from a document should be predictive of that document. After training, the learned word and document representations are used to perform retrieval. Queries are seen as n-grams and matched against documents in the feature space. Documents are then ranked in decreasing order of the cosine similarity computed between query and document representations.

3 Docker Image Architecture

We developed two Docker images reproducing NVSM, one CPU-based and another GPU-based.

The CPU-based version of NVSM is written in Python and relies on `Tensorflow v.1.13.1`. For this reason, we developed a Docker image based on the official

Python 3.5 runtime container, on top of which we install the Python packages required by the algorithm – such as `Tensorflow`, `Python NLTK`, and `Whoosh` – we also install a C compiler, i.e. `gcc`, in order to use the official `trec_eval` package[2] to evaluate the retrieval model during training. Since this docker image still relies for some functions (i.e. random number generation) on the host machine, despite being very similar, the results are not exactly the same across different computers – while they are consistent on the same machine.

The GPU-based version of NVSM is based on `Tensorflow`, which is a machine learning library that allows us to employ the GPU on the host machine in order to perform operations more efficiently. There are many advantages of employing GPUs for scientific computations, but their usage makes a sizable difference especially when training deep learning models. The training of such models requires in fact to perform a large number of matrix operations that can be easily parallelized and do not require powerful hardware.

In our experiments, we observed that `nvsm_gpu` does not produce fully consistent results on the same machine. In fact, `TensorFlow` uses the `Eigen` library, which in turn uses CUDA atomic functions to implement reduction operations, such as `tf.reduce_sum` etc. Those operations are non-deterministic and each operation can introduce small variations. Despite this problem, we still believe that the advantages brought by the usage of a GPU in terms of reduction of computational time – combined with the fact that we detected only very small variations in the *Mean Average Precision at Rank 1000 (MAP)*, *Normalized Discounted Cumulative Gain at Rank 100 (nDCG@100)*, *Precision at Rank 10 (P@10)*, and *Recall* – make this implementation of the algorithm a valid alternative to the CPU-based one.

4 Evaluation

To test our docker image we consider the Robust04 collection, which is composed of TIPSTER corpus Disk 4&5 minus CR. The collection counts 528,155 documents, with a vocabulary of 760,467 different words. The topics considered for the evaluation are topics 301-450, 601-700 from Robust04. Only the field *title* of topics is used for retrieval. The set of topics is split into validation (V) and test (T) sets. Relevance judgments are restricted accordingly. The execution times and memory occupation statistics were computed on an 2018 Alienware Area-51 with an Intel Core i9-7980XE CPU @ 2.60GHz with 36 cores, 64GB of RAM and two GeForce GTX 1080Ti GPUs.

To train the NVSM model, we set the following parameters and hyper-parameters: word representation size $k_w = 300$, number of negative examples $z = 10$, learning rate $\alpha = 0.001$, regularization lambda $\lambda = 0.01$, batch size $m = 51200$, dimensionality of the document representations $k_d = 256$ and n-gram size $n = 16$. We train the model for 15 iterations over the document collection and we select the model iteration that performs best in terms of MAP.

[2] https://github.com/usnistgov/trec_eval.

When comparing the `nvsm_cpu` image and the `nvsm_gpu` image, we observe is that the CPU Docker image takes less space on disk than the GPU one. This is because the former does not need all of the drivers and libraries required by the GPU version of `Tensorflow`. In fact, these libraries make the `nvsm_gpu` image three times larger than the other one.

In Table 1, we report the retrieval results obtained with the two shared Docker images. From these results, we observe that there are small differences, always within ±0.01, between the runs obtained with `nvsm_gpu` on the same machine and with the ones obtained with `nvsm_cpu` on different machines.

	MAP	nDCG@100	P@10	Recall
CPU (run 0)	0.138	0.271	0.285	0.6082
GPU (run 0)	0.137	0.265	0.277	0.6102
GPU (run 1)	0.138	0.270	0.277	0.6066
GPU (run 2)	0.137	0.268	0.270	0.6109

Table 1: Retrieval results on the Robust04 (T) collection computed with the two shared Docker images of NVSM.

The MAP, nDCG@100, P@10, and Recall values obtained with the images are all very similar, and close to the measures reported in the original NVSM paper [9]. Indeed, the absolute difference between the reported MAP, nDCG@100, and P@10 values in [9] and our results is always less than 0.02. As a side note, the MAP values obtained by NVSM are low when compared to the other approaches on Robust04 that can be found in the OSIRRC 2019 library – even 10% lower than some methods that do not apply re-ranking.

In order to further evaluate the performance differences between the runs, we begin computing the RMSE considering the MAP, nDCG@100, and P@10 measures. The RMSE gives us an idea of the performance difference between two runs – averaged across the considered topics. We first compute the average values of MAP, nDCG@100, and P@10 over the three `nvsm_gpu` runs on each topic. Then, we compare these averaged performance measures, for each topic, against the corresponding ones associated to the CPU-based NVSM run we obtained on our machine. These results are reported in Table 2. From the results of this

	NVSM GPU (average)
RMSE (MAP)	0.034
RMSE (nDCG@100)	0.054
RMSE (P@10)	0.140

Table 2: RMSE between the NVSM CPU Docker image and the average of the 3 runs computed with the NVSM GPU Docker image.

evaluation we can observe that the average performance difference across the considered 196 topics is very low when considering the MAP and nDCG@100 measures, while it grows when we consider the top part of the rankings (P@10). In conclusion, the RMSE value is generally low, hence we can confidently say that the models behave in a very similar way in terms of MAP, nDCG@100, and P@10 on all the considered topics.

In Table 3, we report the Kendall's τ measures associated to each pair of runs that we computed. This measure shows us how much the considered rankings are similar to each other. In our case, the runs appear to be quite different from each other, since the Kendall's τ values are all close to 0. In other words, when considering the top 100 results in each run, the same documents are rarely in the same positions in the selected rankings. This result, combined with the fact that the runs achieve all similar MAP, nDCG@100, P@10, and Recall values, leads to the conclusion that the relevant documents are ranked high in the rankings, but are not in the same positions. In other words, NVSM performs a permutation of the documents in the runs, maintaining however the relative order between relevant and non-relevant documents.

	GPU (run 0)	GPU (run 1)	GPU (run 2)	CPU
GPU (run 0)	1.0	0.025	0.025	0.018
GPU (run 1)	0.025	1.0	0.089	0.014
GPU (run 2)	0.025	0.089	1.0	0.009
CPU	0.018	0.014	0.009	1.0

Table 3: Kendall's τ correlation coefficient values between the NVSM GPU and NVSM CPU runs.

5 Final Remarks

In this work, we performed a replicability study of the *Neural Vector Space Model (NVSM)* retrieval model using Docker. First, we presented the architecture and the main functions of a Docker image designed for the replicability of *Neural IR (NeuIR)* models. Secondly, we described the image components and the engineering challenges to obtain deterministic results with Docker using popular machine learning libraries such as `Tensorflow`. We also share two Docker images of the NVSM model: the first, which relies only on the CPU of the host machine to perform its operations, the second, which is able to also exploit the GPU of the host machine, when available.

We observed some differences between the runs computed by the `nvsm_cpu` Docker images on different machines and between the runs computed by the `nvsm_cpu` and `nvsm_gpu` Docker images on the same machine. The differences between `nvsm_cpu` images on different machines are related to the non-determinism of the results, as Docker relies on the host machine for some basic operations which influence the model optimization process through the generation of different pseudo-random number sequences. On the other hand, the differences between `nvsm_gpu` images on the same machine are due to the implementation of some functions in the CUDA and `Tensorflow` libraries. We observed that these operations influence in a sizeable way the ordering of the same documents across different runs, but not the overall distribution of relevant and non-relevant documents in the ranking. Similar differences, that are even more accentuated, can be found between `nvsm_cpu` and `nvsm_gpu` images on the same machine. Therefore, even though these differences may seem marginal in offline evaluation settings, where the focus is on average performance, they are extremely relevant for user-oriented online settings – as they can have a sizeable impact on the user experience and should thus be taken into consideration when deciding whether to use NeuIR models in real-world scenarios.

References

1. Dür, A., Rauber, A., Filzmoser, P.: Reproducing a Neural Question Answering Architecture Applied to the SQuAD Benchmark Dataset: Challenges and Lessons Learned. pp. 102–113 (2018). https://doi.org/10.1007/978-3-319-76941-7_8, https://doi.org/10.1007/978-3-319-76941-7_8
2. Ferro, N., Fuhr, N., Maistro, M., Sakai, T., Soboroff, I.: Overview of CENTRE@CLEF 2019: Sequel in the Systematic Reproducibility Realm. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Tenth International Conference of the CLEF Association (CLEF 2019) (2019)
3. Ferro, N., Marchesin, S., Purpura, A., Silvello, G.: A Docker-Based Replicability Study of a Neural Information Retrieval Model. In: Proc. of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019. vol. 2409, pp. 37–43. CEUR-WS.org (2019), http://ceur-ws.org/Vol-2409/docker05.pdf
4. Freire, J., Fuhr, N., Rauber, A.: Reproducibility of Data-Oriented Experiments in e-Science (Dagstuhl Seminar 16041). Dagstuhl Reports **6**(1), 108–159 (2016). https://doi.org/10.4230/DagRep.6.1.108, http://drops.dagstuhl.de/opus/volltexte/2016/5817
5. Marchesin, S., Purpura, A., Silvello, G.: Focal Elements of Neural Information Retrieval Models. An Outlook through a Reproducibility Study. Information Processing & Management **in print**, 34 (2019)
6. Marchesin, S., Purpura, A., Silvello, G.: A neural vector space model implementation repository (2019), https://github.com/giansilv/NeuralIR/
7. Sakai, T., Ferro, N., Soboroff, I., Zeng, Z., Xiao, P., Maistro, M.: Overview of the NTCIR-14 CENTRE Task. In: Proceedings of the 14th NTCIR Conference on Evaluation of Information Access Technologies. Tokyo, Japan (2019)
8. Soboroff, I., Ferro, N., Sakai, T.: Overview of the TREC 2018 CENTRE Track. In: The Twenty-Seventh Text REtrieval Conference Proceedings (TREC 2018) (2018)
9. Van Gysel, C., de Rijke, M., Kanoulas, E.: Neural Vector Spaces for Unsupervised Information Retrieval. ACM Trans. Inf. Syst. **36**(4), 38:1–38:25 (2018)