Luandri: a Clean Lua Interface to the Indri Search Engine

Bhaskar Mitra*
Microsoft, University College London
Cambridge, UK
bmitra@microsoft.com

Fernando Diaz
Microsoft
New York, USA
fdiaz@microsoft.com

Nick Craswell
Microsoft
Bellevue, USA
nickcr@microsoft.com

ABSTRACT

In recent years, the information retrieval (IR) community has witnessed the first successful applications of deep neural network models to short-text matching and ad-hoc retrieval. It is exciting to see the research on deep neural networks and IR converge on these tasks of shared interest. However, the two communities have less in common when it comes to the choice of programming languages. Indri, an indexing framework popularly used by the IR community, is written in C++, while Torch, a popular machine learning library for deep learning, is written in the light-weight scripting language Lua. To bridge this gap, we introduce Luandri (pronounced "*laundry*"), a simple interface for exposing the search capabilities of Indri to Torch models implemented in Lua.

CCS CONCEPTS

•Information systems →Information retrieval; Web searching and information discovery; •Computing methodologies →*Neural networks;*

KEYWORDS

Information retrieval, application programming interface, neural networks

ACM Reference format:
Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Luandri: a Clean Lua Interface to the Indri Search Engine. In *Proceedings of ACM SIGIR conference, Tokyo, Japan, August 2017 (Under review for SIGIR'17),* 3 pages.
DOI: 10.475/123_4

1 INTRODUCTION

In recent years, deep neural networks (DNNs) have demonstrated early positive results on a variety of standard information retrieval (IR) tasks, including on short-text matching [10, 11, 14, 19, 21, 22] and ad-hoc retrieval [9, 17], and shown promising performances on exciting novel retrieval tasks such as multi-modal retrieval [15] and conversational IR [28, 30]. However, the two research communities focused on neural networks and on IR may have less in common when it comes to the choice of programming languages for implementing their respective toolsets. Popular neural network toolkits are often implemented in (or have bindings for) scripting languages, such as Python[1] (e.g., TensorFlow [1], Theano [2], CNTK [29], Caffe [13], MXNet [3], Chainer [25], and PyTorch[2]) or Lua [12] (e.g., Torch [4]) because of their rapid prototyping capabilities. In contrast, many of the popular indexing frameworks for IR are implemented in C++ (e.g., Indri [24]) or Java (e.g., Terrier [18] and Apache Lucene [16]) putting more emphasis on the speed of execution at runtime. The open-source community has developed Python wrappers over the Indri [26] and the Apache Lucene [20] programming interfaces to expose the functionalities of these rich IR libraries to the programming language. However, there is still a gap that remains to be bridged for non-Python based deep learning toolkits, such as Torch.

Torch[3] is a numeric computing framework popular among the deep neural network community. It has been shown to be significantly faster compared to other toolkits such as TensorFlow on convolutional neural networks in multi-GPU environment [23]. It is implemented using the light-weight scripting language Lua.[4] In this paper, we introduce Luandri (pronounced "*laundry*") – a Lua wrapper over the Indri search engine. In particular, Luandri exposes parts of the Indri query environment application programming interface (API) for document retrieval including support for the rich Indri query language.

2 MOTIVATION

There are a variety of scenarios in which a DNN model can benefit from having access to a search engine during training and/or evaluation. Existing DNN models for ad-hoc retrieval [9, 17], for example, operate on query-document pairs to predict relevance. Running these models on the full corpus is prohibitively costly – therefore the evaluation of these models is often limited to re-ranking top-N candidate documents retrieved by a traditional IR model or a search engine. Typically, these candidate sets are retrieved offline in a process separate from the one in which the DNN is evaluated. However, if the search engine is accessible in the same language as the one in which the DNN is implemented, then the candidate generation step and the DNN-based re-ranking step can follow each other within the same process – removing the requirement to store large quantity of intermediate datasets containing the candidates to be ranked.

DNN models train on labelled data, although in some cases labels can be inferred rather than explicit. For example, many DNN models for IR [9–11, 14, 22] use negative training examples that are sampled uniformly from the corpus. Recently Mitra et al. [17] reported that training with judged negative documents can yield better NDCG performance than training with uniform random negatives. Having

*The author is a part-time PhD student at UCL.

Under review for SIGIR'17, Tokyo, Japan

DOI: 10.475/123_4

[1] https://www.python.org/
[2] https://github.com/pytorch/pytorch
[3] https://github.com/torch/torch7
[4] https://www.lua.org

Code snippet 1: Sample Lua code for searching an Indri index using the Luandri API. The Indri index builder application is used for generating the index beforehand. The search query is written using the popular INQUERY structured operators that are supported natively by Indri for specifying matching constraints. The runQuery method in the Luandri API accepts the request as a Lua table and automatically converts it into the appropriate C++ request object that Indri natively expects. Similarly, the result object returned by Indri in C++ is automatically converted to a Lua table.

```
local luandri = paths.dofile('luandri.lua')
local query_environment = QueryEnvironment()
query_environment:addIndex("path_to_index_file")

local request = {
      query = '#syn( #od1(neural networks) #od1(deep learning)) #greater(year 2009)',
      resultsRequested = 10
                }
local results = query_environment:runQuery(request).results

for k, v in pairs(results) do
    print(v.docid .. '\n' .. v.documentName .. '\n' .. v.snippet .. '\n')
end
```

access to a search engine during training could enable additional methods for generating negative samples, such as using documents that are retrieved by the engine but at lower ranks.

The lack of adequate labelled data available for training DNN models for ad-hoc retrieval has been a focus for the neural IR community [5]. It is possible that alternate strategies for supervision may be considered for training these deep models – including reinforcement learning [27] and training under adversarial settings [8] – which could also make use of retrieval from a full corpus during the model training.

Diaz et al. [6] demonstrated a different application of the traditional retrieval step in the neural IR model. Given a query, they retrieve a set of documents using Indri and use that to train a brand new distributed representation of words specific to that query at run time. Such models, with query-specific representation learning, can be implemented and deployed more easily if the machine learning framework has access to a search engine.

Finally, Ghazvininejad et al. [7] proposed to "lookup" external repositories of facts as part of solving larger tasks using neural network models. Empowering DNN models with access to a search engine may be an exciting area for future exploration.

In all these scenarios, it is useful for a search engine, such as Indri, to be accessible from the same programming language used to implement the DNN. Therefore, we are optimistic that by publicly releasing the Luandri API we will stimulate novel explorations from IR researchers already familiar with Torch.

3 QUERYING INDRI FROM LUA

Indri is an open-source search engine available with the Lemur toolkit.[5] Indri consists of two primary components – an application that builds an index from a raw document collection and another application that can perform searches using this index. The Indri index builder can deal with several different document formats for indexing. This includes TREC (text and Web), HTML, XML, PDF, and plain text among many others.

Searching using Indri involves specifying one or more indices and querying them by either interactively calling the API or by running an application in batch-mode. The Indri query language supports a rich set of operators for specifying phrasal matching conditions, synonymy relationships, document filtering criteria, and other complex constraints. The full query language grammar is available online for reference.[6]

Invoking a search on an Indri index using the Luandri API is very similar to how one may use the native C++ Indri API. Code snippet 1 shows a minimal example of a typical Indri-based search using the Luandri API. We observe that the search is performed by invoking very few lines of Lua code.

The example also demonstrates the use of Indri structured queries. A search is performed using a structured query that constraints the matching to either of the two ordered phrases – "neural networks" or "deep learning". The query directs Indri to treat both phrases as synonyms. In addition, a numeric filter is specified to limit matches to only documents whose value corresponding to the year field is greater than 2009.

This example shows searching on the full document index. However, Luandri also allows users to specify a list of document identifiers in the request object to limit the search to only those set of documents. A fixed list of stop words can also be specified for retrieval using the Luandri API.

The full Luandri implementation is available on GitHub[7] under the MIT license. We direct interested readers to the source code for exact API specifications.

[5] http://www.lemurproject.org/indri/

[6] https://www.lemurproject.org/lemur/IndriQueryLanguage.php

[7] https://github.com/bmitra-msft/Luandri

4 UNDER THE HOOD

The implementation of Lua as a programming language puts a strong emphasis on extensibility [12]. Lua is an *extension language* because any Lua code can be relatively easily embedded as libraries into code written in other languages. It is also an *extensible language* because of its ability to call functions written in other languages, such as C. The implementation of the Luandri API benefits from the latter property of the language.

Lua comes with a fast Just In Time (JIT) compiler called LuaJIT.[8] LuaJIT exposes a foreign-function interface[9] (FFI) that makes it easy to call external C functions and manipulate C data structures from Lua. The Luandri API is written using the LuaJIT FFI library.

Luandri API wraps Indri's query environment data types and methods by extern C functions. Then using the LuaJIT's FFI library these C methods are exposed to any code written in Lua. Luandri automatically handles any conversions necessary between Lua tables and Indri's C++ objects, and vice versa. The "Luandri.cpp" and "luandri.lua" files contain all the wrapper logic on the C++ and the Lua side of our API code, respectively.

The current Luandri API exposes only some of the data structures and methods from Indri's query environment. In future, we hope to expose more of Indri's retrieval functionalities prioritizing based on the need of the broader research community.

5 CONCLUSIONS

We introduced Luandri, a Lua API to the Indri search engine. Luandri brings to DNN models, implemented on Torch, the retrieval capabilities of Indri, including its powerful query language grammar. We posit that the capabilities of a search engine may be useful for training future DNN models for IR – for sampling negative examples, or for training under reinforcement or adversarial settings. We hope that the release of Luandri will not only help researchers working on Torch models for IR, but also stimulate new research in novel DNN models that incorporate retrieval from an external knowledge base as an intermediate step towards solving larger tasks.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, and others. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* (2016).

[2] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: A CPU and GPU math compiler in Python. In *Proc. 9th Python in Science Conf.* 1–7.

[3] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274* (2015).

[4] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

[5] Nick Craswell, W Bruce Croft, Jiafeng Guo, Bhaskar Mitra, and Maarten de Rijke. 2016. Report on the SIGIR 2016 Workshop on Neural Information Retrieval (Neu-IR). 50, 2 (2016), 96–103.

[6] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query Expansion with Locally-Trained Word Embeddings. *arXiv preprint arXiv:1605.07891* (2016).

[7] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. 2017. A Knowledge-Grounded Neural Conversation Model. *arXiv preprint arXiv:1702.01932* (2017).

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[9] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *Proc. CIKM*. ACM, 55–64.

[10] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *Proc. NIPS*. 2042–2050.

[11] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proc. CIKM*. ACM, 2333–2338.

[12] Roberto Ierusalimschy, Luiz Henrique De Figueiredo, and Waldemar Celes Filho. 1996. Lua-an extensible extension language. *Softw., Pract. Exper.* 26, 6 (1996), 635–652.

[13] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 675–678.

[14] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *Advances in Neural Information Processing Systems*. 1367–1375.

[15] Lin Ma, Zhengdong Lu, Lifeng Shang, and Hang Li. 2015. Multimodal convolutional neural networks for matching image and sentence. In *Proceedings of the IEEE International Conference on Computer Vision*. 2623–2631.

[16] Michael McCandless, Erik Hatcher, and Otis Gospodnetic. 2010. *Lucene in Action: Covers Apache Lucene 3.0.* Manning Publications Co.

[17] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2016. Learning to Match Using Local and Distributed Representations of Text for Web Search. *arXiv preprint arXiv:1610.08136* (2016).

[18] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Christina Lioma. 2006. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of the OSIR Workshop*. 18–25.

[19] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition. In *Proc. AAAI*.

[20] Andreas Schreiber. 2009. Mixing Python and Java. (2009).

[21] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proc. SIGIR*. ACM, 373–382.

[22] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Gregoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proc. CIKM*. ACM, 101–110.

[23] Shaohuai Shi, Qiang Wang, Pengfei Xu, and Xiaowen Chu. 2016. Benchmarking State-of-the-Art Deep Learning Software Tools. *arXiv preprint arXiv:1608.07249* (2016).

[24] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. 2005. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, Vol. 2. Citeseer, 2–6.

[25] Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of workshop on machine learning systems (LearningSys) in the twenty-ninth annual conference on neural information processing systems (NIPS)*.

[26] Christophe Van Gysel, Evangelos Kanoulas, and Maarten de Rijke. 2017. Pyndri: a Python Interface to the Indri Search Engine. *arXiv preprint arXiv:1701.00749* (2017).

[27] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[28] Rui Yan, Yiping Song, and Hua Wu. 2016. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*. ACM, 55–64.

[29] Dong Yu, Adam Eversole, Mike Seltzer, Kaisheng Yao, Zhiheng Huang, Brian Guenter, Oleksii Kuchaiev, Yu Zhang, Frank Seide, Huaming Wang, and others. 2014. *An introduction to computational networks and the computational network toolkit.* Technical Report. Tech. Rep. MSR, Microsoft Research, 2014, http://codebox/cntk.

[30] Xiangyang Zhou, Daxiang Dong, Hua Wu, Shiqi Zhao, R Yan, D Yu, Xuan Liu, and H Tian. 2016. Multi-view response selection for human-computer conversation. *EMNLPfi16* (2016).

[8] http://luajit.org

[9] http://luajit.org/ext_ffi.html