Parrot: A Python-based Interactive Platform for Information Retrieval Research

Xinhui Tu
School of Computer Science, Central China Normal University
Wuhan, China
tuxinhui@mail.ccnu.edu.cn

Jimmy Huang
School of Information Technology, York University
Toronto, Canada
jhuang@yorku.ca

Jing Luo
College of Computer Science and Technology, Wuhan University of Science and Technology, China
luojing@wust.edu.cn

Runjie Zhu
School of Information Technology, York University
Toronto, Canada
sherryzh@cse.yorku.ca

Tingting He
School of Computer Science, Central China Normal University
Wuhan, China
tthe@mail.ccnu.edu.cn

ABSTRACT

Open source softwares play an important role in information retrieval research. Most of the existing open source information retrieval systems are implemented in Java or C++ programming language. In this paper, we propose Parrot[1], a Python-based interactive platform for information retrieval research. The proposed platform has mainly three advantages in comparison with the existing retrieval systems: (1) It is integrated with Jupyter Notebook, an interactive programming platform which has proved to be effective for data scientists to tackle big data and AI problems. As a result, users can interactively visualize and diagnose a retrieval model; (2) As an application written in Python, it can be easily used in combination with the popular deep learning frameworks such as Tersorflow and Pytorch; (3) It is designed especially for researchers. Less code is needed to create a new retrieval model or to modify an existing one. Our efforts have focused on three functionalists: good usability, interactive programming, and good interoperability with the popular deep learning frameworks. To confirm the performance of the proposed system, we conduct comparative experiments on a number of standard test collections. The experimental results show that the proposed system is both efficient and effective, providing a practical framework for researchers in information retrieval.

CCS CONCEPTS

• **Information systems → Information retrieval**.

KEYWORDS

information retrieval, deep learning, Python

[1]https://github.com/IR-Community/Parrot or http://www.yorku.ca/jhuang/parrot

ACM Reference Format:
Xinhui Tu, Jimmy Huang, Jing Luo, Runjie Zhu, and Tingting He. 2019. Parrot: A Python-based Interactive Platform for Information Retrieval Research. In *42nd Int'l ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3331184.3331393

1 INTRODUCTION AND MOTIVATION

Over the past several decades, many open source softwares have been built to facilitate the research of information retrieval, such as Okapi [7][8][15], Indri[2], Galago[3], Terrier[4], and Anserini [18]. These softwares greatly improve the efficiency of the evaluation of retrieval models on standard test collections. Most of the existing open source information retrieval systems are implemented in Java or C++ programming language. With the advance of information technology, researchers expect that the IR softwares can keep up with the new requirements, such as good usability, interactive programming, and good interoperability with the popular deep learning frameworks.

Most of the existing retrieval systems such as Terrier and Lucene[5] are over-abstraction. The codes of a retrieval model may be scattered across a number of classes. Users have to understand the complex relationship between the classes before they implement a new retrieval model or modify an existing one.

The interactive programming platforms such as Jupyter Notebook[6] have proved to be effective for data scientists to tackle big data and AI problems. Project Jupyter supports interactive data science and scientific computing via the development of open-source software. The scientific publication Nature has released an article on the advantages of Jupyter Notebooks for scientific research [16]. Compared with the dynamic programming languages like Python, static programming languages such as Java and C++ are not well supported by Jupyter Notebook.

Over the past decade, deep learning has achieved remarkable success in a wide range of domains, such as speech recognition, computer vision, and natural language processing [10]. Over the

[2]https://www.lemurproject.org/indri/
[3]https://www.lemurproject.org/galago.php
[4]http://terrier.org
[5]http://lucene.apache.org/
[6]https://jupyter.org/

Listing 1: The example of how to implement BM25 model

```
from parrot import *

ap90 = datasets.ap90.load("/trec/ap90")

class BM25Model(Model):
  def __init__(self, k1 = 1.2, b = 0.75):
    self.k1 = k1; self.b = b

  def pre_compute(self, df, ctf, ctn, C, N):
    self.avgdl =  C / N
    self.idf = log(1 + (N - df + 0.5) / (df + 0.5))

  def score_term(self, tf, ctn, dl):
    b = self.b; k1 = self.k1
    tf_part = tf * (k1 + 1)
      / (tf + k1 * (1 - b + b * dl / self.avgdl))
    return tf_part * self.idf

result_list = []
for k in frange(0, 1, 0.1):
    model = BM25Model(k1 = 1.2, b = k)
    result = model.run(dataset = ap90)
    result_list.append(result)
```

past few years, many researchers tried to explore deep learning based approaches to information retrieval [12]. It has been proved that neural ranking models are very effective when enough training data is provided [13]. Deep learning frameworks such as Pytorch[7] and Tensorflow[8] provide an effective way to implement a neural ranking model. Because most of the popular deep learning frameworks are written in the Python programming language, it is not easy to integrate them with the existing Java or C++ based retrieval systems. As a popular Python-based project, MatchZoo [3] has implemented a lot of deep text matching models. However, it don't provide necessary functions to conduct all experimental steps of ad-hoc retrieval.

Motivated by the above considerations, we propose Parrot, a Python-based interactive platform for information retrieval research. We aim not only to provide an interactive computing platform for researches to diagnose and visualize retrieval models interactively but also to bridge the gap between the popular deep learning frameworks and the retrieval system. Parrot is built on top of PyLucene[9], which is a Python extension for accessing Java Lucene. Our system provides a lot of intuitive APIs which allow researchers to write less code to create a new model or to modify an existing one. Our efforts have focused on three functionalists: good usability, interactive programming, and good interoperability with the popular deep learning frameworks. Furthermore, Parrot ships with a lot of example codes for standard TREC retrieval tasks, providing a convenient way for researchers to run the baselines.

The existing retrieval models can be classified into three categories: hand-crafted models, learning to rank models, and neural ranking models. Hand-crafted models, such as TF-IDF [17], BM25

[7]https://pytorch.org/

[8]https://www.tensorflow.org/

[9]http://lucene.apache.org/pylucene/

Listing 2: The example of how to use DRMM model

```
from parrot import *

mq2007 = datasets.letor_mq2007.load("/trec/mq2007")

drmm = DRMMModel()
drmm.build()
drmm.cross_validation(dataset = mq2007, epochs = 30)
```

[1][19], and LM [9][14], are very simple and easy to understand. These models integrate three major variables to determine the relevance of a query-document pair: within-document term frequency, document length, and the specificity of the term in the collection [4]. Learning to rank models employ machine learning techniques over hand-crafted IR features [11]. In contrast to classical learning to rank models, more recently proposed neural ranking models directly learn text representations or matching functions [2][5][13]. Both learning to ranking models and neural ranking models require large-scale training data before they can be deployed. Parrot provides a robust framework to run the different types of retrieval models.

To confirm the efficiency of the proposed system, we conducted comparative experiments on a number of standard test collections. The experimental results show that Parrot has better searching and indexing performance than Indri[10], a popular open source retrieval tool written in C++ programming language. Parrot is a good choice for researchers who prefer to program in Python. It provides a practical framework for researchers in information retrieval.

2 MAIN COMPONENTS

Parrot includes three major components: data preparation, retrieval model, and evaluation and diagnosis. The purpose of the data preparation component is to build the index and convert different types of data files into standard form files. Retrieval model component aims to provide a lot of classic retrieval models, including most classic hand-crafted models and a number of neural ranking models. The component of evaluation and diagnosis provides a number of functions to evaluate and diagnose retrieval models.

2.1 Data preparation

Different types of data are required to run different types of retrieval models. To run a hand-crafted retrieval model, we need the data as follows.

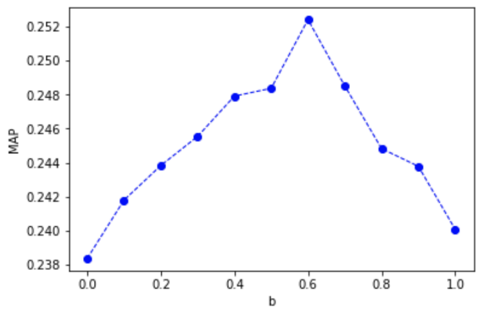
- **Inverted Index** is a data structure storing a mapping from content, such as words or numbers, to its locations in a document. The indexing module first converts different types of the document into the standard form document, and then builds an inverted index.
- **Topic and Judgment** files are used to evaluate the performance of a retrieval model. The topic and judgment parsers extract information from the original TREC files and convert them into standard form files.

[10]https://www.lemurproject.org/indri/

Listing 3: The example code of how to use the evaluation function to generate a MAP figure

```
plot_single_trend(
  x_list = frange(0, 1, 0.1),
  y_list = result_list,
  x_label="b",
  y_label="MAP")
```

Figure 1: The precision change of the BM25 model with different parameter b



Different from the hand-crafted models, neural ranking models need a lot of training data which may be prepared manually by human assessors. The following data are needed to run a neural ranking model.

- **Word Dictionary** lists the words which are used as the input of neural ranking models. We use predefined rules to filter out words that are too frequent, too rare or noisy.
- **Corpus File** records the original text of the collection, which is used to learn representations of language.
- **Training and test data** are used to train and evaluate a ranking model. Typically, we separate a data set into the training set and the testing set. We use most of the data for training and a smaller portion of the data for testing.

Parrot provides a convenient way for researchers to prepare the datasets for the standard test collections. It ships with a lot of example codes for standard TREC retrieval tasks, including AP90, DISK12, DISK45, WT2G, WT10G, and GOV2. List 1 and List 2 show the example codes for data preparation.

2.2 Retrieval model

Parrot is designed especially for researchers in information retrieval. It provides intuitive APIs for them to access the information which is required to build a retrieval model.

Most of the hand-crafted ranking models exploit three parts, such as term frequency, inverse document frequency, and document length normalization, to score a query-document pair [4]. The significant difference between these models is how they combine the three factors. In Parrot, we provide a base class to rank a set of documents for a query. As a result, researchers only need to focus on how to score a term when they implement a new retrieval model. List 1 shows an example of how to implement the BM25 model.

Listing 4: The example of how to conduct a case diagnosis

```
explain_case(
  result = result_list[0],
  doc = 0)
```

In recent years, neural ranking models have become a significant focus in information retrieval. The experimental results indicate that they achieve better performance in comparison with the existing methods [6]. As a software written in Python, Parrot has excellent interoperability with the popular deep learning frameworks such as Tersorflow and Pytorch, which will facilitate the development of neural ranking models. Because MatchZoo has implemented a lot of deep text matching models include a number of popular neural ranking models, we don't need to reinvent the wheel. We provides a unified data preparation module for different neural ranking models. List 2 shows an example of how to use the DRMM [13] model for ranking.

2.3 Evaluation and diagnosis

The component of evaluation and diagnosis provides a lot of functions for researchers to evaluate and diagnose a retrieval model, which is very important for a retrieval system designed for researchers. Researchers can use these functions in an interactive way under Jupyter notebook.

This component includes two modules: the evaluation module and the diagnosis module. Evaluation module provides several widely used metric such as MAP and NDCG to evaluate the performance of a retrieval model. It can output the TREC-compatible ranking results. It also provides functions for users to interactively visualize the performance of a retrieval model. Parameter sensitivity analysis is a commonly used method to find the optimal parameter value for a retrieval model. List 3 shows the example code of how to generate a MAP figure, which is depicted in Figure 1. The figure shows the precision change of the BM25 model with different parameter values.

Sometimes, a ranking model may give a low score to a relevant document. Researchers need to study the scoring details to figure out why the ranking model fails. Diagnosis module provides case diagnose functions for users to investigate the bad cases of a ranking result. List 4 shows an example of how to conduct a case diagnosis.

3 PERFORMANCE EVALUATION

To confirm the performance of our system, we conducted comparative experiments on several standard test collections. We conducted the experiments on a workstation with Xeon processors running Ubuntu.

Invert indexing is an essential component of a retrieval system. In dealing with large document collections, operational efficiency is necessary for a retrieval system. Parrot is built based on PyLucene, a Python extension for accessing Java Lucene. In this paper, we conducted indexing experiments on the five standard test collections. Table 1 shows the indexing performances of Parrot and Indri. The experimental results show that Parrot is faster than Indri in term of indexing time.

Table 1: Indexing performance of Parrot and Indri

			Indri		Parrot(pos)		Parrot(doc)	
Collection	docs	terms	time	size	time	size	time	size
Disk12	742k	129m	00:12:03	2.5GB	00:01:34	519MB	00:03:02	2.5GB
Disk45	528k	175m	00:06:42	1.9GB	00:01:28	432MB	00:02:39	2.1GB
WT2G	246k	182m	00:06:58	2.2GB	00:02:41	442MB	00:04:07	2.4GB
WT10G	1.69m	752m	00:39:51	9.6GB	00:04:53	3.0GB	00:09:16	12GB
Gov2	25.2m	17.3b	13:26:07	215GB	02:28:36	11GB	06:08:29	337GB

Table 2: Retrieval efficiency on Gov2, using a single process

	Latency (ms)	throughput (qps)
Indri	2003	0.52
Parrot	341	2.96

Table 3: : Effectiveness comparisons between Parrot and Indri on standard TREC test collections

Collection	Disk12	Disk45	WT2G	WT10G	Gov2
Queries	51-200	301-450 601-700	401-450	451-550	701-850
BM25 (I)	0.2040	0.2478	0.3152	0.1955	0.2970
BM25 (P)	0.2296	0.2514	0.3145	0.2009	0.3051
LM (I)	0.2269	0.2516	0.3116	0.1915	0.2995
LM (P)	0.2281	0.2502	0.2985	0.2104	0.3011

To verify the search efficiency of Parrot, we also conducted search experiments on the Gov2 collection. We issued 100,000 queries sequentially against both the Parrot and Indri indexes. Table 2 lists the experimental results, including latency (ms) and throughput (queries per second, or qps). The experimental results demonstrate that Parrot is more efficient than Indri in terms of search performance.

Finally, we compared the retrieval effectiveness of Parrot and Indri. We conducted the experiments on five standard TREC collections. Table 3 shows the experimental results, where (I) refers to Indri and (P) refers to Parrot. The results indicate that Parrot and Indri achieve comparable performance in terms of mean precision.

4 CONCLUSION AND FUTURE WORK

In this paper, we have presented a Python-based toolkit to perform IR experimentation within Jupyter notebook. As mentioned above, we believe this will help us in (1) providing an interactive experimental framework for IR; (2) facilitating the development of neural ranking models, and (3) improving the efficiency of the research in information retrieval. In the future, we will implement classical learning-to-rank models.

ACKNOWLEDGMENTS

This work is substantially supported by the National Natural Science Foundation of China under the grant number 61572223. This work is also supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada, an NSERC CREATE award in ADERSIM, the York Re-search Chairs program, and an Ontario Research Fund-Research Excellence award in BRAIN Alliance. We thank anonymous reviewers for their thorough review comments.

REFERENCES

[1] M Beaulieu, M Gatford, Xiangji Huang, S Robertson, S Walker, and P Williams. 1997. Okapi at TREC-5. *NIST SPECIAL PUBLICATION SP* (1997), 143–166.
[2] Yixing Fan, Jiafeng Guo, Yanyan Lan, Jun Xu, Chengxiang Zhai, and Xueqi Cheng. 2018. Modeling Diverse Relevance Patterns in Ad-hoc Retrieval. *arXiv preprint arXiv:1805.05737* (2018).
[3] Yixing Fan, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2017. Matchzoo: A toolkit for deep text matching. *arXiv preprint arXiv:1707.07270* (2017).
[4] Hui Fang, Tao Tao, and Chengxiang Zhai. 2011. Diagnostic evaluation of information retrieval models. *ACM Transactions on Infor. Sys.* 29, 2 (2011), 7.
[5] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* ACM, 55–64.
[6] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. 2019. A Deep Look into Neural Ranking Models for Information Retrieval. *arXiv preprint arXiv:1903.06902* (2019).
[7] Xiangji Huang and Qinmin Hu. 2009. A bayesian learning approach to promoting diversity in ranking for biomedical information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.* ACM, 307–314.
[8] Xiangji Huang, Fuchun Peng, Dale Schuurmans, Nick Cercone, and Stephen E Robertson. 2003. Applying machine learning to text segmentation for information retrieval. *Information Retrieval* 6, 3-4 (2003), 333–362.
[9] John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 111–119.
[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436.
[11] Tie-Yan Liu et al. 2009. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.
[12] Bhaskar Mitra and Nick Craswell. 2017. Neural Models for Information Retrieval. *arXiv preprint arXiv:1705.01509* (2017).
[13] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management.* ACM, 257–266.
[14] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 275–281.
[15] Stephen E Robertson. 1997. Overview of the okapi projects. *Journal of documentation* 53, 1 (1997), 3–7.
[16] Helen Shen. 2014. Interactive notebooks: Sharing the code. *Nature News* 515, 7525 (2014), 151.
[17] Amit Singhal, Gerard Salton, Mandar Mitra, and Chris Buckley. 1996. Document length normalization. *Infor. Processing & Management* 32, 5 (1996), 619–633.
[18] Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: Enabling the Use of Lucene for Information Retrieval Research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 1253–1256.
[19] Jiashu Zhao, Jimmy Xiangji Huang, and Ben He. 2011. CRTER: using cross terms to enhance probabilistic information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval.* ACM, 155–164.