Dockerizing Indri for OSIRRC 2019

Claudia Hauff
Delft University of Technology
Delft, The Netherlands
c.hauff@tudelft.nl

ABSTRACT

The Lemur Project was set up in 2000 by the Center for Intelligent Information Retrieval at UMass Amherst. It is one of the longest lasting open-source projects in the information retrieval (IR) research community. Among the released tools is `Indri`, a popular search engine that was designed for language-modeling based approaches to IR. For OSIRRC 2019 we dockerized `Indri` and added support for the `Robust04`, Core18 and `GOV2` test collections.

1 OVERVIEW

As part of the Lemur Project[1] a number of tools have been developed, most notably `Galago`, `Indri` [4], and `RankLib`. `Indri` has been—and still is—a widely popular research search engine implemented in C++ which allows for the efficient development and evaluation of novel language-modeling based approaches to IR. In addition, `Indri` offers a query language that provides support for constraints based on proximity, document fields, syntax matches, and so on.

We here describe the implementation of the `Indri` Docker image[2] for the OSIRRC 2019 challenge, the incorporated baselines, results and issues observed along the way.

2 DOCKER IMAGE DESIGN

The design of our Docker image is tied to the jig,[3] a toolkit developed specifically for OSIRRC 2019, which provides a number of "hooks" (such as `index` and `search`) that are particular to the workflow of search systems.

2.1 Dockerfile

The Dockerfile builds an image based on Ubuntu 16.04. Apart from `Indri` v5.13 itself, a number of additional software package are installed such as `nodejs` (one of the scripts to prepare the Core18 collection is a Node.js script) and `python` (to interact with the jig).

2.2 `index`

This hook indexes the corpora mounted by the jig, making use of `Indri`'s `IndriBuildIndex`. We support three corpora (Core18, `GOV2` and `Robust04`), which each require different preprocessing steps:

`Robust04` The original `Robust04` corpus is `.z` compressed, a compression format `Indri` does not support. And thus, we first need to uncompress the corpus and filter out undesired folders such as `cr` (as `Indri` does not support excluding particular subfolders from indexing) before starting the indexing process.

Core18 The corpus is provided in JSON format and first needs to be converted to a document format `Indri` supports.

`GOV2` Among the three corpora only `GOV2` is well suited for `Indri`, it can be indexed without any further preprocessing.

The created indices are stemmed (Krovetz) with stopwords removed. For the latter, we relied on the Lemur project stopword list[4] which contains 418 stopwords.

```
<top>
<num> Number: 301
<title> International Organized Crime

<desc> Description:
Identify organizations that participate in international
criminal activity, the activity, and, if possible,
collaborating organizations and the countries involved.

<narr> Narrative:
A relevant document must as a minimum identify the
organization and the type of illegal activity (e.g.,
Columbian cartel exporting cocaine). Vague references to
international drug trade without identification of the
organization(s) involved would not be relevant.
</top>
```

Figure 1: TREC topic 301.

2.3 `search`

This hook is responsible for creating a retrieval run.

Topic files. In a preprocessing step, the TREC topic files (an example topic of `Robust04` is shown in Figure 1) have to be reformatted as `Indri` requires topic files to adhere to a particular format.

Next to reformatting, special characters (punctuation marks, etc.) have to be removed. `Indri` does not provide specific tooling for this step, and one either has to investigate how exactly `Indri` deals with special characters during the indexing phase (thus matching the processing of special characters in order to achieve optimal retrieval effectiveness) or rely on very restrictive filtering (removing anything but alphanumeric characters). We opted for the latter. In contrast, stemming does not have to be applied, as `Indri` applies the same stemming to each query as specified in the index manifest (creating during the indexing phase).

Only standard stopword removal is applied to the topics; this means that in the TREC description and TREC narrative phrases

[1] https://www.lemurproject.org/
[2] https://github.com/osirrc/indri-docker/
[3] https://github.com/osirrc/jig
[4] http://www.lemurproject.org/stopwords/stoplist.dft

```
<query>
<number>301-LM</number>
<text>#combine( international organized crime )</text>
</query>

<query>
<number>301-SD</number>
<text>#weight( 0.9 #combine(international organized crime)
0.05 #combine(#1(international organized) #1(organized crime) )
0.05 #combine(#uw8(international organized) #uw8(organized crime)) )</text>
</query>

<query>
<number>301-DESC</number>
<text>#combine( identify organizations that participate in international criminal activity the activity and
if possible collaborating organizations and the countries involved )</text>
</query>

<query>
<number>301-BM25</number>
<text>international organized crime</text>
</query>
```

Figure 2: Overview of query formats `Indri` accepts. While the language-modeling based approaches can make use of `Indri`'s query language, `Indri`'s baselines tfidf and bm25 (last example query) cannot.

such as *Identify ...* or *A relevant document ...* (cf. Figure 1) remain in the final query after preprocessing.

Moreover, different retrieval methods require differently formatted topic files (e.g. the BM25 retrieval model does not support complex queries, cf. Figure 2). Depending on the topic type (e.g., TREC title-only topics, description-only topics, title+description topics) different queries are created.

Retrieval rules. The jig provides an option `--opts` which allows extra options to be passed to the search hook. We use it, among others, to specify (i) the retrieval rule,[5] (ii) whether to include pseudo-relevance feedback (PRF, `use_prf="1"`) and (iii) whether to use the sequence dependency (SD, `sd="1"`) model. The hyperparameters for both PRF and SD are fixed. Specifically, for PRF we use 50 feedback documents, 25 feedback terms and equal weighting of the original and expanded query model. The SD weights are set to 0.9 for the original query, 0.05 for bigrams and 0.05 for unordered windows. These settings were based on prior works. A better approach would be to employ hyperparameter tuning.

3 RESULTS

Table 1 showcases the use of the optional parameter of the jig's `search` hook to set the retrieval rules. We report the retrieval effectiveness in MAP. When comparing our results to those reported in prior works using `Indri` and (at least) `Robust04` [1–3, 5] we report similar trends, though with smaller absolute effectiveness differences: SD and PRF are both more effective than the vanilla language modeling approach and their combination performs best. BM25 performs somewhat worse than expected, an outcome we argue is due to our lack of hyperparameter tuning. The biggest differences can be found in the results we report for queries solely derived from the TREC topic descriptions (instead of a combination of title and description): our results are significantly worse than the title-only baseline, which we attribute to a lack of "cleaning up" those descriptions (i.e. removing phrases like *Relevant documents include*).

4 CONCLUSIONS

Creating the Docker image for `Indri` was more work than anticipated. One unexpected problem turned out to be the sourcing of the original corpora (instead of processed versions suited for `Indri` that had been "passed down" from researcher to researcher within our lab). In addition, for almost every corpus/topic set combination a different preprocessing script had to be written which turned into a lengthy process as (i) `Indri` tends to fail silently (e.g. a failure to process a query with special characters will only be flagged when running `trec_eval` as the exception is simply written to the result file) and (ii) debugging a Docker image is not trivial.

In the next step, we will implement automatic hyperparameter tuning.

ACKNOWLEDGEMENTS

This research has been supported by NWO project SearchX (639.022.722).

[5] All retrieval methods as documentend at https://lemurproject.org/doxygen/lemur/html/IndriRunQuery.html are supported.

	Robust04	GOV2	Core18
`--opts out_file_name="outfile" rule="method:dirichlet,mu:1000" topic_type="title"`	0.2499	0.2800	0.2332
`--opts out_file_name="outfile" rule="method:dirichlet,mu:1000" topic_type="title" sd="1"`	0.2547	0.2904	0.2428
`--opts out_file_name="outfile" rule="method:dirichlet,mu:1000" topic_type="title" use_prf="1"`	0.2812	0.3033	0.2800
`--opts out_file_name="outfile" rule="method:dirichlet,mu:1000" topic_type="title" use_prf="1" sd="1"`	**0.2855**	**0.3104**	**0.2816**
`--opts out_file_name="outfile" rule="okapi,k1:1.2,b:0.75" topic_type="title+desc"`	0.2702	0.2705	0.2457
`--opts out_file_name="outfile" rule="method:dirichlet,mu:1000" topic_type="desc"`	0.2023	0.1336	0.1674

Table 1: Overview of the optional parameter settings of the search hook and the corresponding retrieval effectiveness as measured in MAP.

REFERENCES

[1] Michael Bendersky, Donald Metzler, and W Bruce Croft. 2012. Effective query formulation with multiple information sources. In *Proceedings of the fifth ACM international conference on Web search and data mining*. ACM, 443–452.

[2] Zhuyun Dai and Jamie Callan. 2019. Deeper Text Understanding for IR with Contextual Neural Language Modeling. *arXiv preprint arXiv:1905.09217* (2019).

[3] Van Dang and Bruce W Croft. 2010. Query reformulation using anchor text. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 41–50.

[4] Trevor Strohman, Donald Metzler, Howard Turtle, and W Bruce Croft. 2005. Indri: A language model-based search engine for complex queries (extended version). CIIR Technical Report.

[5] Guoqing Zheng and Jamie Callan. 2015. Learning to reweight terms with distributed representations. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. ACM, 575–584.