Entity Retrieval Docker Image for OSIRRC at SIGIR 2019

Negar Arabzadeh
narabzad@ryerson.ca
Ryerson University
Toronto, Ontario

ABSTRACT

With emerging of structured data, retrieving entities instead of documents becomes more prevalent in order to satisfy the information need related to a query. Therefore, several high-performance entity retrieval methods have been introduced to the Information Retrieval (IR) community in recent years. Replicating and reproducing the standard entity retrieval methods are considered as challenging tasks in the IR community. Open-Source IR Replicability Challenge (OSIRRC) has addressed this problem by introducing a unified framework for dockerizing a variety of retrieval tasks. In this paper, a Docker image is built for six different entity retrieval models including, LM, MLM-tc, MLM-all, PRMS, SDM, FSDM. Also, Entity Linking incorporated Retrieval(ELR) extension, has been implemented that can be applied on top of all the mentioned models. The entity retrieval docker can retrieve relevant entities for any given topic.

Image Source: https://github.com/osirrc/entityretrieval-docker
Docker Hub: https://hub.docker.com/r/osirrc2019/entityretrieval

1 OVERVIEW

In the past two decades, search engines have been dealing with unorganized and unclassified data i.e., unstructured data until the emergence of semantic search. In order to satisfy the information need behind a query using structured data, retrieving machine-recognizable "entities" has proven to be a suitable complementary for document retrieval for multiple reasons. For instance, Returning a document in response to a query that is looking for an entity might not be the best option because users have to look into the document to find their desired information need. That could be one of the main reason why retrieval operations are getting more and more entity-centric, particularly on the web. Document retrieval differs from entity retrieval in a couple of senses. In document retrieval, entities are usually used for query expansion or retrieval features in order to improve learning-to-rank frameworks and consecutively to enhance document retrieval performance. On the other hand, entity retrieval is defined as searching for an entity in a knowledge base where entities are first class citizens[2]. In other words, our goal is to retrieve the most relevant entities from a knowledge base for a given term-based query.

In this docker image, the following standard entity retrieval models have been implemented:

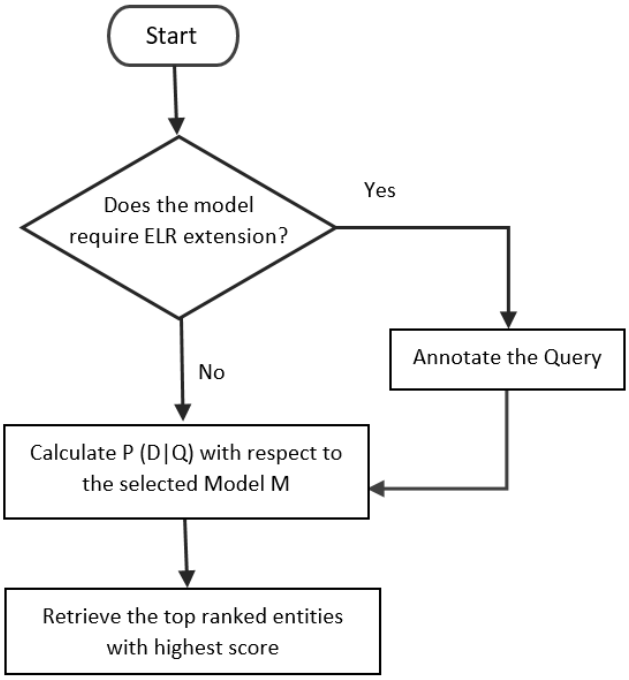

Figure 1: entity retrieval flowchart for a given query Q and retrieval model M. D is the relevant required representation of entities for the model M.

- LM [9]
- MLM-tc [7]
- MLM-all [8]
- PRMS [5]
- SDM [6]
- FSDM [10]

Furthermore, an extension of the Markov Random Field (MRF) model framework for incorporating entity annotations into the retrieval model, which is called Entity linking incorporated Retrieval(ELR) has been leveraged on top of mentioned retrieval model. Applying ELR on the state-of-the-art entity retrieval models results in having the following ELR-integrated entity retrieval models [2] :

- LM_{elr}
- $MLM - tc_{elr}$
- $MLM - all_{elr}$
- $PRMS_{elr}$
- SDM_{elr}
- $FSDM_{elr}$

DBpedia version 3.9 has been used as the knowledge base in the entity retrieval tasks in this Docker image. A term-based index and an entity-based index had created from the knowledge base by utilizing Lucene. Within the former index, entities URI objects are resolved to terms and the default Lucene stop words have been

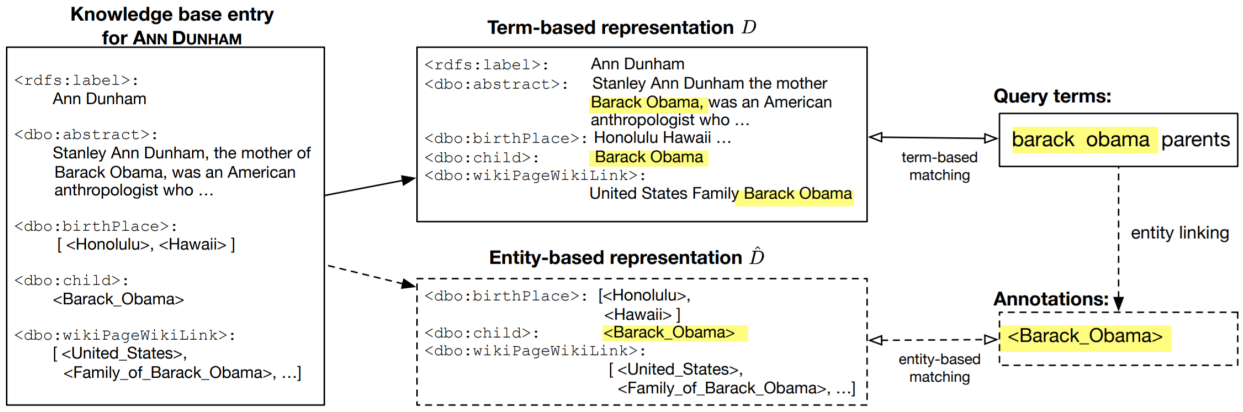

Figure 2: Term-based VS Entity-based representation [2]

removed from them. Meanwhile, within the latter, URI objects are preserved and literal objects are ignored. In total, entity-based index contains 3,984,580 entities. While the term-based index is used in the standard entity retrieval model, both term based and entity based indices are used in ELR entity retrieval models.

One of the critical components of the ELR approaches is entity annotations of the queries. TAGME, which is an open source entity linker, has been adopted to annotate entities in queries with the default threshold 0.1. Hasibi et al.[2] have shown that the ELR approach is robust to annotation threshold.

In summary, given a query Q and a retrieval model M, the entity retrieval operates according to Figure 1, where D is the representation of entities. In section 2, there will be a more in-depth elaboration on standard entity retrieval methods in addition to the combination of them with ELR extension. Section 3 provides more details on the technical design aspect of the docker image. Section 4 describes our motivation and experience in participating in the OSIRRC 2019 challenge. The last section, i.e., section 5, gives an insight on further work that has to be carried out in this area and conclude the paper.

2 RETRIEVAL MODELS

As was mentioned in the Overview section, several standard retrieval models including LM, MLM-tc, MLM-all, PRMS, SDM and FSDM have been implemented in this docker. In addition, ELR extension can be applied on top of them, which results in having twelve different retrieval models collectively. There will be two different representation of entities; the term-based representation and the entity-based representation. For the standard retrieval models, term-based representation of DBpedia collection (using term-based index) is used and when it comes to ELR extension. On the other hand, both term-based and entity-based representation of entities in DBpedia are used (term-based and URI-based indices). The term-based and entity-based representations are compared in Figure 2 [2].

Retrieval models can be categorized into Language modeling-based models and Sequential Dependence models and ELR models.

2.1 Language Modeling-based methods

Language modeling-based models consider dependencies among query terms. LM [9], MLM-tc[7], MLM-all[8] and PMRS[5] are all language modeling based methods. LM only applies on the `content` field. However, MLM-tc run against `name` field as well as `content` field . The fields `content` and `name` have weights of 0.8 and 0.2 respectively. MLM-all and PRMS use top 10 fields . The former is a Mixture of Language models with top 10 fields with uniform weights but the latter's retrieval model is a probabilistic one designed for semi-structured data. More details on each of the mentioned language models can be found in the related papers.

2.2 Sequential Dependence-based methods

Sequential Dependence Models (SDM) are popular Markov Random Field based retrieval models. Given a document D and a query Q, the conditional probability of P(D|Q) is estimated based on Markov Random Field as in equation (1) where D is term-based representation of entities.

$$P(D|Q) \overset{\text{rank}}{=} \sum_{c \in C(G)} \lambda_c f(c) \tag{1}$$

In equation (1), C(G) is set of cliques in graph G . The nodes of the graph G consist of query terms and documents and the edges among nodes illustrates the dependencies among nodes. λ_c is weight of the feature function *f(c)* . More details can be found in the original paper [6].

Considering dependencies among query terms results in having equation (2) based on Markov Random Field (equation (1)) as SDM ranking function with respect to $\lambda_T + \lambda_O + \lambda_U = 1$:

$$\begin{aligned} P(D|Q) \overset{\text{rank}}{=} & \lambda_T \sum_{q_i \in Q} f_T(q_i, D) + \\ & \lambda_O \sum_{q_i, q_{i+1} \in Q} f_O(q_i, q_{i+1}, D) + \\ & \lambda_U \sum_{q_i, q_{i+1} \in Q} f_U(q_i, q_{i+1}, D)\} \end{aligned} \tag{2}$$

2.2.1 Fielded Sequential Dependence Models (FSDM) .

Fielded Sequential Dependence Models (FSDM) considers document structure by computing linear interpolation of probability of each documents' fields.Thus, feature functions are also calculated based on field representation of documents. In other words, in FSDM model [10] equation (2) comes with different feature functions as different language models are built for each field.

2.3 ELR models

Incorporating *Entity Linking into entity Retrieval* leads to improve entity retrieval performance [2]. Linking entities by TAGME results in having confidence score $s(e)$ for each entity e. While considering sequential dependency in MRF-based models, annotated queries are assumed to be independent of each other and query terms. Applying ELR extension on the previous models results in the equation(3) as ranking function where $|Q|$ is the query length and $s(e)$ is the entity linking confidence score of entity e annotated by TAGME. Equation(3) is elaborated more in [6] with its feature functions. Free parameters constraints of $\lambda_T + \lambda_O + \lambda_U + \lambda_E = 1$ is true for equation 3. For LM, MLM-tc and MLM-all λ_O and λ_U are set to zero since they are unigram based models. All the feature functions are defined in [2].

$$
\begin{aligned}
P(D|Q) \overset{\text{rank}}{=} & \lambda_T \sum_{q_i \in Q} \frac{1}{|Q|} f_T(q_i, D) + \\
& \lambda_O \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_O(q_i, q_{i+1}, D) + \\
& \lambda_U \sum_{q_i, q_{i+1} \in Q} \frac{1}{|Q|-1} f_U(q_i, q_{i+1}, D) + \\
& \lambda_E \sum_{e \in E(Q)} s(e) f_E(e, D)\}
\end{aligned}
\tag{3}
$$

3 TECHNICAL DESIGN

One of the major issues when dealing with replicability problem, is that the system should be delivered in a lightweight package[1]. Dockers has this ability since a relatively inexpensive container can be created from each docker image. *a jig* was introduced in OSIRRC2019 workshop that makes the co-implementing and co-designing available. The jig which is open source and available on GitHub[1] plays a semi-tool role which can maintain computational relationship among Dockers and retrieval tasks.

The entity retrieval Docker image is consisted of `init`, `index` and `search` hooks invokes by Python3 as its interpreter. The jig triggers the `init` hook first and then`index` and `search` respectively in the Docker image. Finally , if there golden standard for the topics are available, it will evaluate the results. Since we can get data into and out of the container built from the Docker image [1], we can get the retrieval results in the jig output directory. further explanation to run the entity retrieval Docker image is available on the entity retrieval GitHub repository.

[1]https://github.com/osirrc/jig

This section describes different components of the docker image and supported hooks and extra options which can be passed to the jig for the entity retrieval Docker.

3.1 Dockerfile

The latest official version of Ubuntu[2] is installed in the Docker with all the required commands. In addition, compatible versions of other requirements such as java8, Apache Ant, Apache Ivy, g+, and so on are installed on the Docker. Making all the components compatible with each other was a quite challenging issue. Since installing all the requirements is a time-consuming step, a docker image is prepared with all the basic requirements and pushed it to Docker Hub[3]. Hence, this prepared image is used as our Dockerfile base image in order to decrease the building time of the Docker. This sets the stage for COPYing the init,index and search hooks which should be executable files.

3.2 Supported Collections

DBpedia version 3.9 [4] has been used as the corpus of entity retrieval Docker image. In order to reduce the run time cost of the docker, the original index is used. Both term-based indexed and URI-based indexed of the collection will be downloaded once in the preparation step of the jig. However, to make the docker run using "the jig", a dummy collection is needed to pass.

3.3 Supported Hooks

This section elaborates on the role of each hook separately. `init` and `index` hooks are triggered in the jig preparation step and `search` hook script will run in the jig search step.

3.3.1 init.

The actual implementation of the retrieval models is cloned in this hook from the GitHub repository[5]. The required compatible packages are installed. Running this hook may take a while because of downloading,building and installing PyLucene which is time-consuming. Once the installation are done, two indexed collection which are DBpedia term-based index and URI-based index are downloaded ($\sim 18GB$) and extracted.

3.3.2 index.

The indexed DBpedia collection is already downloaded in the `init` hook. Hence, nothing is happening in this hook in this docker.

3.3.3 search.

When the image is prepared, retrieval models can run with respect to their relevant customized parameters in the `search` hook. The `search` hook runs the main implementation of models[6] which were provided by Hasibi et al.[2]. the main code was cloned in `init` hook in the Docker. Table 1 demonstrates all the parameters that can be set for each of the retrieval models. Given the query , depending on the retrieval model, the query would be annotated or not, and then the retrieval takes place based on the set parameters. Then,

[2]https://hub.docker.com/_/ubuntu
[3]https://hub.docker.com/r/narabzad/elr_prepared_os
[4]https://wiki.dbpedia.org/services-resources/datasets/data-set-39/downloads-39
[5]https://github.com/Narabzad/elr_files
[6]https://github.com/hasibi/EntityLinkingRetrieval-ELR/

Table 1: Entity retrieval models acceptable parameters which are entity linking threshold (*threshold*), number of selected fieleds (*nfields*) and free paramaters ($\lambda_T, \lambda_O, \lambda_U, \lambda_E$). For each model, parameters with ✓ affect the retrieval model and × indicates that parameter does not have any affect on the model. According to each model, Some parameters might have been set to zero.

	threshold	nfields	λ_T	λ_O	λ_U	λ_E
LM	×	×	✓	0	0	×
MLM-tc	×	×	✓	0	0	×
MLM-all	×	✓	✓	0	0	×
PRMS	×	✓	✓	0	0	×
SDM	×	×	✓	✓	✓	×
FSDM	×	✓	✓	✓	✓	×
LM_{ELR}	✓	×	✓	0	0	✓
MLM-tc_{ELR}	✓	×	✓	0	0	✓
MLM-all_{ELR}	✓	✓	✓	0	0	✓
$PRMS_{ELR}$	✓	✓	✓	0	0	✓
SDM_{ELR}	✓	×	✓	✓	✓	✓
$FSDM_{ELR}$	✓	✓	✓	✓	✓	✓

the ranked list of retrieved entities for each query will be saved in the `output` repository.

All the queries in the jig e.g, topics of `Robust04`, `ClueWeb09`, `ClueWeb12`, `Gov2`, `Core17`, `Core18` and etc. are supported in this Docker. Any other queries is acceptable in this docker as long as each query is represented in the following format "query number or name:query terms" in each line. An instance of a topic file would be like:

```
wt09-1:obama family tree
wt09-2:french lick resort and casino
wt09-3:getting organized
...
```

If relevant entities (*qrel*) are available for the associated topics, the jig will utilize Trec Eval [7] to evaluate the retrieval performance by different metrics such as MAP. If there is no ground truth ranked list of entities for the topic, a dummy qrel file is needed to pass to the jig in order to make the Docker run by the jig.

4 OSIRRC EXPERIENCE

The crucial role of repeatability, replicability, and reproducibility cannot be neglected in any research domain; especially when it comes to practical experiments. Deciding to participate in this challenge was easy because either it is repeating your computation, replicating another researchers' experiments or reproducing other team's research with a totally different setup, there will be an endeavored struggle. This docker tackles all these 3 challenges for entity retrieval. Not only the docker is built by a non-author of the main paper (replicability) [2], but also this work is not limited to specific topics. In other words, the entity retrieval docker is modified in a way that any topics can be used to retrieve the relevant entities (reproducibility) and if the relevant entities i.e., the golden standard, is available for the topics, the model can be evaluated as well. However, the supported collection is still limited to the indexed DBpedia version 3.9.

Dockerizing the entity retrieval models was a challenging task. furthermore, standardizing the Docker with the jig increased its complexity. One of the main issues regarding this Docker was the compatibility of different components e.g, Python, PyLucene, Java, Apache Ant, etc. It was a time-consuming task to find all the compatible version of all those components and this is one of the critical benefits of this task. Utilizing the entity Docker, Researchers do not have to spend lots of time on combining and connecting different packages, libraries and components anymore to run the mentioned entity retrieval models.

Another issue is that topics appear in different formats. we must be able to work with every topic format available in the jig. Therefore a standard topic format is defined in section 3.3.3 so that any topic can be used in this Docker.

For the methods with ELR extension, the annotation step has to be added to the code that was implemented by Hasibi et al. [2]. Utilizing TAGME tool results in linking entities to queries.

5 FUTURE WORKS AND CONCLUDING REMARKS

The more reproducible and replicable research papers are, the more baselines will be accessible for researches to compare their results with. This means, by increasing repeatability, reproducibility,and replicability researchers can can spend less time on implementing other researchers work. Therefore, they will have more time spending on their own research and not on implementing the baselines. Consecutively, studies would make progress faster. Hence, more works needed to be done in this specific area.

According to entity retrieval docker image, this work can be extended by supporting more collection such as DBpedia-entity v2 [4] in addition to the current one. Nordlys [3] implements some of the models with the updated collection. So in the future, a Docker could be created for Nordlys, which provides better support for indexing.

Furthermore, entity retrieval models can be added to the Docker. In terms of entity retrieval applications, it can be used to expand queries in order to improve document retrieval performance.

To sum up our work, Docker image is wrapped around the jig introduced for Open-Source IR Replicability Challenge 2019. This platform provides a unified framework for different retrieval task. Entity retrieval Docker image contains implementation of six different entity retrieval model. ELR extension also can be applied on any of the models. All models can be customized with desired parameters and there is no limit in the supported topics. This docker image is implemented based on very lightweight Linux-centric design to tackle repeatability, reproducibility and replicability problem in the IR domain.

ACKNOWLEDGEMENT

Thanks to Faegheh Hasibi for her valuable suggestions during the implementation of the Docker image and preparation of this paper.

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

REFERENCES

[1] Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu. 2019. The SIGIR 2019 Open-Source IR Replicability Challenge (OSIRRC 2019)+. https://doi.org/10.1145/3331184.3331647

[2] Faegheh Hasibi, Krisztian Balog, and Svein Erik Bratsberg. 2016. Exploiting Entity Linking in Queries for Entity Retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12- 6, 2016*. 209–218. https://doi.org/10.1145/2970398.2970406

[3] Faegheh Hasibi, Krisztian Balog, Darío Garigliotti, and Shuo Zhang. 2017. Nordlys: A Toolkit for Entity-Oriented and Semantic Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 1289–1292. https://doi.org/10.1145/3077136.3084149

[4] Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. 2017. DBpedia-Entity v2: A Test Collection for Entity Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*. 1265–1268. https://doi.org/10.1145/3077136.3080751

[5] Jinyoung Kim, Xiaobing Xue, and W. Bruce Croft. 2009. A Probabilistic Retrieval Model for Semistructured Data. In *Advances in Information Retrieval, 31th European Conference on IR Research, ECIR 2009, Toulouse, France, April 6-9, 2009. Proceedings*. 228–239. https://doi.org/10.1007/978-3-642-00958-7_22

[6] Donald Metzler and W. Bruce Croft. 2005. A Markov random field model for term dependencies. In *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*. 472–479. https://doi.org/10.1145/1076034.1076115

[7] Robert Neumayer, Krisztian Balog, and Kjetil Nørvåg. 2012. When Simple is (more than) Good Enough: Effective Semantic Search with (almost) no Semantics. In *Advances in Information Retrieval - 34th European Conference on IR Research, ECIR 2012, Barcelona, Spain, April 1-5, 2012. Proceedings*. 540–543. https://doi.org/10.1007/978-3-642-28997-2_59

[8] Paul Ogilvie and James P. Callan. 2003. Combining document representations for known-item search. In *SIGIR 2003: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 28 - August 1, 2003, Toronto, Canada*. 143–150. https://doi.org/10.1145/860435.860463

[9] ChengXiang Zhai. 2008. Statistical Language Models for Information Retrieval: A Critical Review. *Foundations and Trends in Information Retrieval* 2, 3 (2008), 137–213. https://doi.org/10.1561/1500000008

[10] Nikita Zhiltsov, Alexander Kotov, and Fedor Nikolaev. 2015. Fielded Sequential Dependence Model for Ad-Hoc Entity Retrieval in the Web of Data. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*. 253–262. https://doi.org/10.1145/2766462.2767756