An Extensible Toolkit of Query Refinement Methods and Gold Standard Dataset Generation

Hossein Fani[1(✉)], Mahtab Tamannaee[2], Fattane Zarrinkalam[2], Jamil Samouh[2], Samad Paydar[2], and Ebrahim Bagheri[2]

[1] School of Computer Science, University of Windsor, Windsor, Canada
hfani@uwindsor.ca

[2] Laboratory for Systems, Software and Semantics (LS3), Ryerson University, Toronto, Canada
{mtamannaee,fzarrinkalam,jsamouh,paydar,bagheri}@ryerson.ca

Abstract. We present an open-source extensible python-based toolkit that provides access to a (1) range of built-in unsupervised query expansion methods, and (2) pipeline for generating gold standard datasets for building and evaluating supervised query refinement methods. While the information literature offers abundant work on query expansion techniques, there is yet to be a tool that provides unified access to a comprehensive set of query expansion techniques. The advantage of our proposed toolkit, known as ReQue (refining queries), is that it offers one-stop shop access to query expansion techniques to be used in external information retrieval applications. More importantly, we show how ReQue can be used for building gold standards datasets that can be used for training supervised deep learning-based query refinement techniques. These techniques require sizeable gold query refinement datasets, which are not available in the literature. ReQue provides the means to systematically build such datasets.

1 Introduction

To improve retrieval performance, query refinement methods fill the gap between the language of the user's query and that of the relevant information by formulating an alternative set of terms for the original query either through an unsupervised approach, e.g., adding more synonyms with similar significance (see [2] for a comprehensive study), or via a supervised learning approach that learns how to reformulate the original query q to a refined version q' from a set of labeled training samples $(q \rightarrow q')$, e.g., neural-based models that have recently received more attention [6,11,19].

While the literature has extensively explored methods of supervised and unsupervised query expansion, there are two major limitations in this area: (1) while the implementation of some supervised query expansion methods are sporadically available by the authors, the implementation of many others are not

D. Hiemstra et al. (Eds.): ECIR 2021, LNCS 12657, pp. 498–503, 2021.
https://doi.org/10.1007/978-3-030-72240-1_54

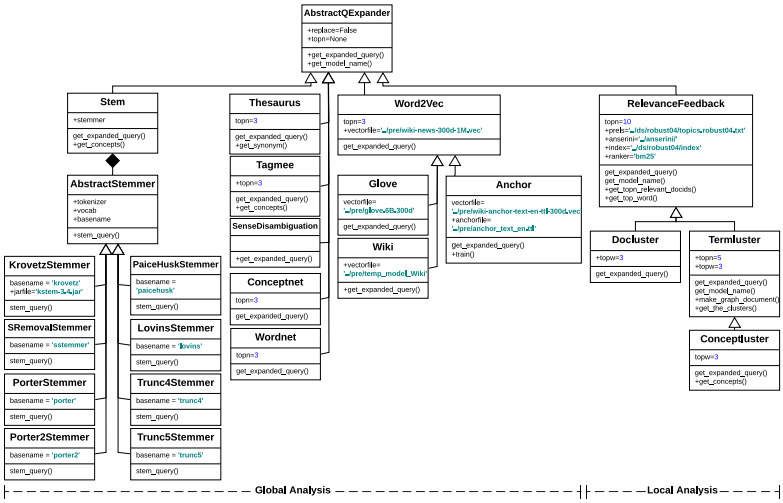


Fig. 1. Inheritance hierarchy for expanders available in ReQue.

available. The implementation of those available are also not compatible with each other and require substantial revision to be deployable; and (2) even though supervised query refinement methods require gold standard dataset for training, there is no known *truly* gold standard dataset for this purpose. Such methods are often benchmarked on session-based search history of the users such as aol [17] and msmarco [15] where the main underlying assumption is that the last query in the users' search session is a better reformulation of the original query and can be considered to be an acceptable *refined query*. The expectation is that the user has gradually revised her query over successive attempts to find relevant information within the same session. This assumption however has not been confirmed neither empirically nor theoretically in the literature. Indeed, intuitive counterexamples can be easily provided from real-world user search sessions as extracted from the msmarco dataset in [20]. In order to address these two major limitations, we make a python-based extensible toolkit, called ReQue, publicly available that offers the following capabilities:

1. Like similar efforts in the community such as MatchZoo [5], which offer an extensible platform of the design, comparison and sharing of deep text matching models, ReQue offers a platform to publicly share query expansion techniques. It comes with a host of implemented query expansion methods and offers an object-oriented structure that can easily facilitate and accommodate the addition of new query expansion methods;
2. Based on its set of built-in unsupervised query expansion methods, ReQue offers a pipeline to automatically generate gold standard datasets to be used for training supervised query refinement methods. This is a major advantage as most state of the art neural query refinement methods are now developed

```python
# ./qe/expanders/abstractqexpander.py
class AbstractQExpander:
  def __init__(self,replace=False,topn=None):
    self.replace = replace
    self.topn = topn
    def get_expanded_query(self,q,args=None):
      return q
# ./qe/stemmers/abstractstemmer.py
class AbstractStemmer():
  def __init__(self): ...
  def stem_query(self,q): ...
```

(a)

```python
# ./qe/expanders/relevancefeedback.py
class RelevanceFeedback(AbstractQExpander):
  def __init__(self, ranker, ..., topn=10):...
  def get_expanded_query(self, q, args): ...
  def get_topn_relevant_docids(self, qid): ...
  def get_top_word(self, tfidf): ...

# ./qe/expanders/stem.py
from stemmers.abstractstemmer import AbstractStemmer
class Stem(AbstractQExpander):
  def __init__(self, stemmer:AbstractStemmer):
    AbstractQExpander.__init__(self)
    self.stemmer = stemmer
    def get_expanded_query(self,q,args=None):
      return self.stemmer.stem_query(q)
```

(b)

Fig. 2. ReQue's code snippets for implementing expanders.

based on *silver* standard query datasets, which we have already shown to include considerable limitations [20].

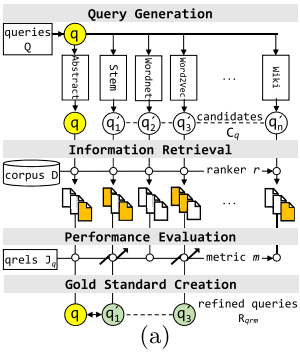
ReQue has currently integrated a host of 21 state-of-the-art unsupervised query expansion methods including, but not limited to, lexical [18], semantic [7,16,21], embedding [9], corpus-based clustering [3,14], web-based [1,8,12], and pseudo-relevance feedback [10] methods. The methods are shown in the inheritance hierarchy of classes in Fig. 1 and can be classified into *global* and *local* analysis methods. The codebase along with the installation instructions and video tutorials as well as case studies on trec topics can be obtained at https://github.com/hosseinfani/ReQue/tree/ecir2021demo.

Demonstration. During the presentation of this work, we will (1) introduce the built-in query expansion methods that are offered by ReQue and how they can be used in other applications, (2) show how new query expansion methods can be easily integrated into ReQue, (3) demonstrate the workflow provided by ReQue for generating gold standard datasets to train and evaluate supervised query refinement methods [4,13,19], and (4) review the statistical characteristics of the gold standard datasets that are generated by ReQue.

2 Toolkit Overview

2.1 Query Expansion in ReQue

ReQue offers a set of query expansion methods, called *expanders*, to generate expanded queries for any input query. At its core, and as shown in Fig. 2a, ReQue includes the *identity* expander, called `AbstractQExpander`, as the abstract root of the class hierarchy whose main method `get_expanded_query()` is to be overridden by expanders. Adding a new expander to ReQue is as easy as extending an existing expander, and modifying `get_expanded_query()`, as shown in Fig. 2b for `RelevanceFeedback` and `Stem` expanders. The expanders are also able to revise the original query by either adding (`replace=False`) or replacing (`replace=True`) its terms with `topn` new related terms. ReQue currently



(a)

```
# ./qe/main.py
from cmn import expander_factory as ef
def run(op,rf=True,..):

  expanders = ef.get_nrf_expanders()
  if rf:#relevance feedback (local analysis)
    expanders += ef.get_rf_expanders(rankers,..)
  if 'generate' in op:generate(...)
  if 'search'   in op:search(...)
  if 'evaluate' in op:evaluate(...)
  if 'build' in op:
    result = aggregate(...)
    build(input=result,...,output=output)

def generate(Qfilename,expanders,output):
def search(expanders,rankers,index,output,...):
def evaluate(expanders,qrels,rankers,metrics,...):
def aggregate(expanders,rankers,metrics,...)
def build(input,expanders,rankers,metrics,output):
```

(b)

Fig. 3. ReQue's gold standard generation (a) workflow and (b) core engine.

has integrated a host of 21 state-of-the-art unsupervised query expansion methods (*c.f.* Fig. 1). Due to a modular design, the expanders can be easily combined/mixed into new expander mashups.

2.2 Gold Standard Generation in ReQue

ReQue adopts a simple yet effective approach to generate gold standard query refinement datasets on existing query sets such as those offered by trec competitions or msmarco [15]. Reque takes three inputs: 1) a set of queries $Q = \{q\}$ along with their associated relevance judgements (`qrels`) J_q in a corpus D, e.g., `robust04`, 2) an information retrieval method (ranker) r, e.g., `bm25`, and 3) an evaluation metric m, e.g., mean average precision (`map`). As shown in Fig. 3, a host of state-of-the-art expanders are used foremost to systematically generate a large number of revised candidate queries C_q for each original query q. Next, the revised candidate queries C_q are evaluated based on how they improve the performance of the given ranker r in terms of the evaluation metric m given the relevance judgments J_q for corpus D. Finally, given the performances of the revised candidate queries, those that provide better improvement compared to the original query q are selected as the refined queries $R_{qrm} \subseteq C_q$.

Out of the box, ReQue includes gold standard datasets for `robust04`, `gov2`, `clueweb09b` and `clueweb12b13` based on `bm25` and `qld` as the rankers and `map` as the evaluation metric along with benchmark results of 3 supervised query refinement methods [4,13,19]. Statistics including the average number of refined queries and the average `map` improvement rate for each of these gold standard datasets has been reported in [20] and show that for all the rankers, at least `1.44` refined queries exists on average for an original query while the best performance is for `robust04` over `bm25` with `4.24`. Given the best refined query for each original query, the average `map` improvement rate is greater than `100%` for all the gold standard datasets which means the best refined query for an original query almost *doubled* the performance of the ranker in terms of `map`.

3 Concluding Remarks

This paper introduces ReQue that benefits the IR community by the seamless access to query expansion methods as well as to the development of gold standard datasets for the task of supervised query refinement. Key contributions include:

1. ReQue is designed with extensibility in mind. While it already hosts a wide variety of unsupervised query expansion methods, it is quite easy to add new unsupervised query expansion methods, supervised query refinement models, or user-defined query refinement methods to it;
2. ReQue automatically generates gold standard datasets for training and evaluating supervised query refinement methods. It can be easily configured based on an original query set, its associated relevance judgements, a ranker of choice and an evaluation metric. It ensures refined queries improve the performance of the ranker in terms of the evaluation metric;
3. ReQue aids reproducibility and repeatability of the research work on shared gold standard datasets. As a part of its release, it includes gold standard datasets for each of the `robust04`, `gov2`, `clueweb09b` and `clueweb12b13` document collections and their associated trec topics based on `bm25` and `qld` as the rankers and `map` as the evaluation metric.

References

1. Al-Shboul, B., Myaeng, S.-H.: Wikipedia-based query phrase expansion in patent class search. Inf. Retrieval **17**(5–6), 430–451 (2013). https://doi.org/10.1007/s10791-013-9233-4
2. Azad, H.K., Deepak, A.: Query expansion techniques for information retrieval: a survey. Inf. Process. Manag. **56**(5), 1698–1735 (2019)
3. Carpineto, C., de Mori, R., Romano, G., Bigi, B.: An information-theoretic approach to automatic query expansion. ACM Trans. Inf. Syst. **19**(1), 1–27 (2001)
4. Dehghani, M., Rothe, S., Alfonseca, E., Fleury, P.: Learning to attend, copy, and generate for session-based query suggestion. In: 2017 ACM on Conference on Information and Knowledge Management, pp. 1747–1756 (2017)
5. Guo, J., Fan, Y., Ji, X., Cheng, X.: Matchzoo: a learning, practicing, and developing system for neural text matching. In: SIGIR 2019, pp. 1297–1300. ACM, New York (2019)
6. Han, F.X., Niu, D., Chen, H., Lai, K., He, Y., Xu, Y.: A deep generative approach to search extrapolation and recommendation. In: KDD 2019, pp. 1771–1779. ACM (2019)
7. Hsu, M.-H., Tsai, M.-F., Chen, H.-H.: Query expansion with ConceptNet and WordNet: an intrinsic comparison. In: Ng, H.T., Leong, M.-K., Kan, M.-Y., Ji, D. (eds.) AIRS 2006. LNCS, vol. 4182, pp. 1–13. Springer, Heidelberg (2006). https://doi.org/10.1007/11880592_1
8. Kraft, R., Zien, J.Y.: Mining anchor text for query refinement. In: WWW 2004, pp. 666–674. ACM (2004)
9. Kuzi, S., Shtok, A., Kurland, O.: Query expansion using word embeddings. In: CIKM 2016, pp. 1929–1932. ACM (2016)

10. Lee, K., Croft, W.B., Allan, J.: A cluster-based resampling method for pseudo-relevance feedback. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, pp. 235–242. ACM (2008)
11. Li, R., Li, L., Wu, X., Zhou, Y., Wang, W.: Click feedback-aware query recommendation using adversarial examples. In: WWW 2019, pp. 2978–2984. ACM (2019)
12. Li, Y., Zheng, R., Tian, T., Hu, Z., Iyer, R., Sycara, K.P.: Joint embedding of hierarchical categories and entities for concept categorization and dataless classification. In: COLING 2016, pp. 2678–2688. ACL (2016)
13. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. In: 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 1412–1421. The Association for Computational Linguistics (2015)
14. Natsev, A., Haubold, A., Tesic, J., Xie, L., Yan, R.: Semantic concept-based query expansion and re-ranking for multimedia retrieval. In: Proceedings of the 15th International Conference on Multimedia, pp. 991–1000. ACM (2007)
15. Nguyen, T., et al.: MS MARCO: a human generated machine reading comprehension dataset. In: NIPS 2016 (2016)
16. Pal, D., Mitra, M., Datta, K.: Improving query expansion using wordnet. J. Assoc. Inf. Sci. Technol. **65**(12), 2469–2478 (2014)
17. Pass, G., Chowdhury, A., Torgeson, C.: A picture of search. In: Infoscale 2006, p. 1 (2006)
18. Schofield, A., Mimno, D.M.: Comparing apples to apple: the effects of stemmers on topic models. Trans. Assoc. Comput. Linguistics **4**, 287–300 (2016)
19. Sordoni, A., Bengio, Y., Vahabi, H., Lioma, C., Simonsen, J.G., Nie, J.: A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In: CIKM 2015, pp. 553–562. ACM (2015)
20. Tamannaee, M., Fani, H., Zarrinkalam, F., Samouh, J., Paydar, S., Bagheri, E.: Reque: a configurable workflow and dataset collection for query refinement. In: CIKM2020, pp. 3165–3172. ACM (2020)
21. Tan, L.: Pywsd: python implementations of word sense disambiguation (WSD) technologies [software]. https://github.com/alvations/pywsd