Covidex: Neural Ranking Models and Keyword Search Infrastructure for the COVID-19 Open Research Dataset

Edwin Zhang,[1] **Nikhil Gupta,**[1] **Raphael Tang,**[1] **Xiao Han,**[1] **Ronak Pradeep,**[1] **Kuang Lu,**[2] **Yue Zhang,**[2] **Rodrigo Nogueira,**[1] **Kyunghyun Cho,**[3,4] **Hui Fang,**[2] and **Jimmy Lin**[1]

[1] University of Waterloo [2] University of Delaware
[3] New York University [4] CIFAR Associate Fellow

Abstract

We present Covidex, a search engine that exploits the latest neural ranking models to provide information access to the COVID-19 Open Research Dataset curated by the Allen Institute for AI. Our system has been online and serving users since late March 2020. The Covidex is the user application component of our three-pronged strategy to develop technologies for helping domain experts tackle the ongoing global pandemic. In addition, we provide robust and easy-to-use keyword search infrastructure that exploits mature fusion-based methods as well as standalone neural ranking models that can be incorporated into other applications. These techniques have been evaluated in the multi-round TREC-COVID challenge: Our infrastructure and baselines have been adopted by many participants, including some of the best systems. In round 3, we submitted the highest-scoring run that took advantage of previous training data and the second-highest fully automatic run. In rounds 4 and 5, we submitted the highest-scoring fully automatic runs.

1 Introduction

As a response to the worldwide COVID-19 pandemic, on March 13, 2020, the Allen Institute for AI (AI2) released the COVID-19 Open Research Dataset (CORD-19) (Wang et al., 2020). With regular updates since the initial release (first weekly, then daily), the corpus contains around 300,000 scientific articles (as of October, 2020), including most with full text, about COVID-19 and coronavirus-related research more broadly (for example, SARS and MERS). These articles are gathered from a variety of sources, including PubMed, a curated list of articles from the WHO, as well as preprints from arXiv, bioRxiv, and medRxiv. The goal of the effort is "to mobilize researchers to apply recent advances in natural language processing to generate new insights in support of the fight against this infectious disease." We responded to this call to arms.

As motivation, we believe that information access capabilities (search, question answering, etc.) can be applied to provide users with high-quality information from the scientific literature, to inform evidence-based decision making and to support insight generation. Examples include public health officials assessing the efficacy of wearing face masks, clinicians conducting meta-analyses to update care guidelines based on emerging studies, and virologist probing the genetic structure of COVID-19 in search of vaccines. We hope to contribute to these efforts via a three-pronged strategy:

1. Despite significant advances in the application of neural architectures to text ranking, keyword search (e.g., with "bag of words" queries) remains an important core technology. Building on top of our Anserini IR toolkit (Yang et al., 2018), we have released robust and easy-to-use open-source keyword search infrastructure that the broader community can build on.
2. Leveraging our own infrastructure, we explored the use of sequence-to-sequence transformer models for document expansion and candidate reranking, combined with a simple classification-based feedback approach to exploit existing relevance judgments. We have also open sourced all these models, which can be integrated into other systems.
3. Finally, we package the previous two components into Covidex, an end-to-end search engine and browsing interface deployed at covidex.ai, initially described in Zhang et al. (2020a).

All three efforts have been successful. In the TREC-COVID challenge, our infrastructure and baselines have been adopted by many teams, which in some

Proceedings of the First Workshop on Scholarly Document Processing, pages 31–41
Online, November 19, 2020. ©2020 Association for Computational Linguistics
https://doi.org/10.18653/v1/P17

cases have submitted runs that scored higher than our own submissions. This illustrates the success of our infrastructure-building efforts (1). In round 3, we submitted the highest-scoring run that took advantage of previous training data and the second-highest fully automatic run. In rounds 4 and 5, we submitted the highest-scoring fully automatic runs. These results affirm the quality of our own ranking models (2). Finally, usage statistics offer some evidence for the success of our deployed Covidex search engine (3).

2 Ranking Components

Multi-stage search architectures represent the most common design for modern search engines, with work in academia dating back over a decade (Matveeva et al., 2006; Wang et al., 2011; Asadi and Lin, 2013). Known production deployments of this design include the Bing web search engine (Pedersen, 2010) as well as Alibaba's e-commerce search engine (Liu et al., 2017).

The idea behind multi-stage ranking is straightforward: instead of a monolithic ranker, ranking is decomposed into a series of stages. Typically, the pipeline begins with an initial retrieval stage, most often using bag-of-words queries against an inverted index. One or more subsequent stages reranks and refines the candidate set successively until the final results are presented to the user. The multi-stage design provides a clean interface between keyword search, neural reranking models, and the user application.

This section details individual components in our architecture. We describe later how these building blocks are assembled in the deployed system (Section 3) and for TREC-COVID (Section 4.2).

2.1 Keyword Search

In our design, initial retrieval is performed by the Anserini IR toolkit (Yang et al., 2017, 2018),[1] which we have been developing for several years and powers a number of our previous systems that incorporate various neural architectures (Yang et al., 2019; Yilmaz et al., 2019). Anserini represents an effort to better align real-world search applications with academic information retrieval research: under the covers, it builds on the popular and widely-deployed open-source Lucene search library, on top of which we provide a number of missing features for conducting research on modern IR test collections.

[1]http://anserini.io/

Anserini provides an abstraction for document collections, and comes with a variety of adaptors for different corpora and formats: web pages in WARC containers, XML documents in tarballs, JSON objects in text files, etc. Providing keyword search capabilities over CORD-19 required only writing an adaptor for the corpus that allows Anserini to ingest the documents.

An issue that immediately arose with CORD-19 concerned the granularity of indexing, i.e., what should we consider to be a "document" as the "atomic unit" of indexing and retrieval? One complication is that the corpus contains a mix of articles that vary widely in length, not only in terms of natural variations (scientific articles of varying lengths, book chapters, etc.), but also because the full text is not available for some articles. It is well known in the IR literature, dating back several decades (e.g., Singhal et al. 1996), that length normalization plays an important role in retrieval effectiveness.

Guided by previous work on searching full-text articles (Lin, 2009), we explored three separate indexing schemes:

- An index comprised of only titles and abstracts.
- An index comprised of each full-text article as a single, individual document; articles without full text contained only titles and abstracts.
- A paragraph-level index structured as follows: each full-text article was segmented into paragraphs and for *each* paragraph, we created a "document" comprising the title, abstract, and that paragraph. The title and abstract alone comprised an additional "document". Thus, a full-text article with n paragraphs yielded $n + 1$ separate retrieval units in the index.

To be consistent with standard IR parlance, we call each of these retrieval units a "document", in a generic sense, despite their composite structure.

In addition to the above indexing schemes, we considered three more based on our doc2query document expansion technique (Nogueira et al., 2019b; Nogueira and Lin, 2019). The idea behind using document expansion is to enhance each document with (synthetic) queries for which the document may be relevant, to alleviate the vocabulary mismatch problem by increasing the likelihood that query terms and document terms match. We used the T5-base doc2query model trained on the MS MARCO passage dataset (Bajaj et al., 2018) provided by Nogueira and Lin (2019). Due to limited computational resources, we only generated expansions from the article abstracts. However, even the abstracts alone often exceeded the model's input length restriction of 512 tokens. To address this issue, we first segmented each document into passages by applying a sliding window of ten sentences with a stride of five. These passages were then prepended with the title of the article. Inference was performed on these passages using a top-k sampling decoder that generated 40 queries (i.e., expansions) per abstract passage. Finally, for each of the index conditions above, we expanded the documents by appending all the expansion queries to form three more doc2query-enhanced indexes. Note that when applying inference with neural networks during the reranking stage, we used the original abstracts (i.e., without expansions).

In all cases (both the original indexes and doc2query-enhanced indexes), documents were initially retrieved using the popular BM25 scoring function (Robertson et al., 1994).

With the paragraph index, a query is likely to retrieve multiple paragraphs from the same underlying article; since the final task is to rank articles, we took the highest-scoring paragraph of an article as its score. Articles were then ranked according to their scores. Furthermore, we combined results from these different indexing schemes to capture different ranking signals using fusion techniques, which further improved effectiveness; see Section 4.2 for details.

Since Anserini is built on top of Lucene, which is implemented in Java, it is designed to run on the Java Virtual Machine (JVM). However, TensorFlow (Abadi et al., 2016) and PyTorch (Paszke et al., 2019), the two most popular neural network toolkits today, use Python as their main language. More broadly, with its diverse and mature ecosystem, Python has emerged as the language of choice for most data scientists today. Anticipating this gap, we have been working on Pyserini,[2] Python bindings for Anserini, since late 2019 (Yilmaz et al., 2020). Pyserini is released as a well-documented, easy-to-use Python module distributed via PyPI and easily installable via `pip`.[3]

Putting everything together, we provide the community keyword search infrastructure by sharing code, indexes, as well as baseline runs. First, all our code is available open source. Second, we share pre-built versions of CORD-19 indexes, so that users can replicate our results with minimal effort. Finally, we provide baseline runs for TREC-COVID that can be directly incorporated into other participants' submissions.

2.2 Rerankers

In our infrastructure, the output of Pyserini is fed to rerankers that aim to improve ranking quality. We describe three different approaches: two are based on neural architectures, and the third exploits relevance judgments in a feedback setting using a classification approach.

monoT5. Despite the success of BERT for document ranking (Dai and Callan, 2019; MacAvaney et al., 2019; Yilmaz et al., 2019), there is evidence that ranking with sequence-to-sequence models can achieve even better effectiveness, particularly in zero-shot and other settings with limited training data (Nogueira et al., 2020a,b), such as for TREC-COVID. Our "base" reranker, called monoT5, is based on T5 (Raffel et al., 2020).

Given a query q and a set of candidate documents D from Pyserini, for each $d \in D$ we construct the following input sequence to feed into our model:

$$\text{Query: } q \text{ Document: } d \text{ Relevant:} \tag{1}$$

The model is fine-tuned to produce either "true" or "false" depending on whether the document is relevant or not to the query. That is, "true" and "false" are the ground truth predictions in the sequence-to-sequence task, what we call the "target tokens".

At inference time, to compute probabilities for each query–document pair, we apply softmax only to the logits of the "true" and "false" tokens. We rerank the candidate documents according to the probabilities assigned to the "true" token. See Nogueira et al. (2020a,b) for additional details about this logit normalization trick and the effects of different target tokens.

Since in the beginning we did not have training data specific to COVID-19, we fine-tuned our model on the MS MARCO passage dataset (Bajaj et al., 2018), which comprises 8.8M passages obtained from the top 10 results retrieved by the Bing search engine (based on around 1M queries). The training set contains approximately 500K pairs of query and relevant documents, where each query

[2] `http://pyserini.io/`

[3] `https://pypi.org/project/pyserini/`

has one relevant passage on average; non-relevant documents for training are also provided as part of the training data. Nogueira et al. (2020a,b) and Yilmaz et al. (2019) have previously demonstrated that models trained on MS MARCO can be effectively applied to other document ranking tasks in a zero-shot manner.

We fine-tuned our monoT5 model with a constant learning rate of 10^{-3} for 10K iterations with class-balanced batches of size 128. We used a maximum of 512 input tokens and two output tokens (one for the target token, either "true" or "false", and another for the end-of-sequence token). In the MS MARCO passage dataset, none of the inputs required truncation when using this length limit. Training model variants based on T5-base and T5-3B took approximately 4 and 40 hours, respectively, on a single Google TPU v3-8.

At inference time, since output from Pyserini is usually longer than the length restrictions of the model, it is not possible to feed the *entire* text of the document into our model at once. To address this issue, we first segmented each document into passages by applying a sliding window of ten sentences with a stride of five. We obtained a probability of relevance for each passage by performing inference on it independently, and then selected the highest probability among the passages as the relevance score of the document.

duoT5. A pairwise reranker estimates the probability $s_{i,j}$ that candidate d_i is more relevant than d_j for query q, where $i \neq j$. Nogueira et al. (2019a) demonstrated that a pairwise BERT reranker running on the output of a pointwise BERT reranker in a multi-stage ranking pipeline yielded statistically significant improvements in output quality. We applied the same intuition to T5 to build a pairwise reranker called duoT5, which takes as input the following sequence:

Query: q Document0: d_i Document1: d_j Relevant:

where d_i and d_j are unique pairs of candidates from the set D. The model is fine-tuned to predict "true" if candidate d_i is more relevant than d_j to query q and "false" otherwise. We fine-tuned duoT5 using the same hyperparameters as monoT5.

At inference time, we used the top 50 highest-scoring documents according to monoT5 as our candidate set $\{d_i\}$. We then obtained probabilities $p_{i,j}$ of d_i being more relevant than d_j for all unique candidate pairs $\{d_i, d_j\}, \forall i \neq j$. Finally, we computed a single score s_i for candidate d_i as follows:

$$s_i = \sum_{j \in J_i} (p_{i,j} + (1 - p_{j,i})) \qquad (2)$$

where $J_i = \{0 \leq j < 50, j \neq i\}$. Based on exploratory studies on the MS MARCO passage dataset, this setting led to the most stable and effective rankings.

Relevance Feedback. The setup of TREC-COVID (see Section 4.1) provided a feedback setting where systems can exploit a limited number of relevance judgments on a per-query basis. How do we take advantage of such training data? Despite work on fine-tuning transformers in a few-shot setting (Zhang et al., 2020b; Lee et al., 2020), we were wary of the dangers of overfitting on limited data, particularly since there is little guidance on relevance feedback using transformers in the literature. Instead, we implemented a robust approach that treats relevance feedback as a document classification problem using simple linear classifiers, described in Yu et al. (2019) and Lin (2019).

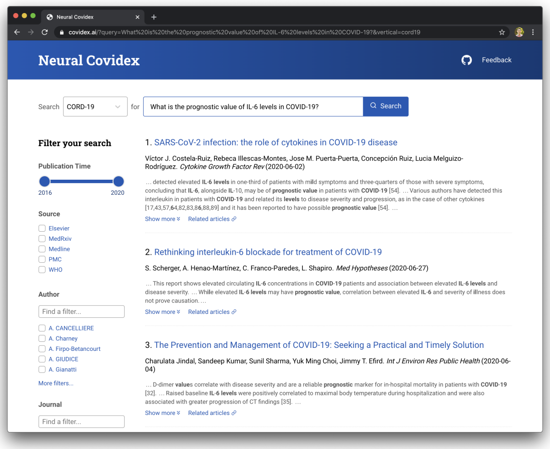
The approach is conceptually simple: for each query, we trained a linear classifier (logistic regression) that attempts to distinguish relevant from non-relevant documents *for that query*. The classifier operated on sparse bag-of-words representations using tf–idf term weighting. At inference time, each candidate document was fed to the classifier, and the classifier score was then linearly interpolated with the original candidate document score to produce a final score. We describe the input source documents in Section 4.2.

All components above have also been open sourced. The two neural reranking modules are available in PyGaggle,[4] which is our recently developed neural ranking library designed to work with Pyserini. Our classification-based approach to feedback is implemented in Pyserini directly. These components are available for integration into any system.

3 The Covidex

Beyond sharing our keyword search infrastructure and reranking models, we've built the Covidex as an operational search engine to demonstrate our capabilities to domain experts who are not interested in individual components. As deployed, we use the paragraph index and monoT5-base as the

[4] http://pygaggle.ai/



1. SARS-CoV-2 infection: the role of cytokines in COVID-19 disease

Víctor J. Costela-Ruiz, Rebeca Illescas-Montes, Jose M. Puerta-Puerta, Concepción Ruiz, Lucia Melguizo-Rodríguez. *Cytokine Growth Factor Rev* (2020-06-02)

… detected elevated **IL-6 levels** in one-third of patients with mild symptoms and three-quarters of those with severe symptoms, concluding that **IL-6**, alongside **IL**-10, may be of **prognostic value** in patients with **COVID-19** [54]. … Various authors have detected this interleukin in patients with **COVID-19** and related its **levels** to disease severity and progression, as in the case of other cytokines [17,43,57,64,82,83,86,88,89] and it has been reported to have possible **prognostic value** [54]. …

2. Rethinking interleukin-6 blockade for treatment of COVID-19

S. Scherger, A. Henao-Martínez, C. Franco-Paredes, L. Shapiro. *Med Hypotheses* (2020-06-27)

… This report shows elevated circulating **IL-6** concentrations in **COVID-19** patients and association between elevated **IL-6 levels** and disease severity. … While elevated **IL-6 levels** may have **prognostic value**, correlation between elevated **IL-6** and severity of illness does not prove causation. …

3. The Prevention and Management of COVID-19: Seeking a Practical and Timely Solution

Charulata Jindal, Sandeep Kumar, Sunil Sharma, Yuk Ming Choi, Jimmy T. Efird. *Int J Environ Res Public Health* (2020-06-04)

… D-dimer **value**s correlate with disease severity and are a reliable **prognostic** marker for in-hospital mortality in patients with **COVID-19** [32]. … Raised baseline **IL-6 levels** were positively correlated to maximal body temperature during hospitalization and were also associated with greater progression of CT findings [35]. …

Figure 1: Screenshot of the Covidex.

reranker. An additional highlighting module based on BioBERT was described in Zhang et al. (2020a). To decrease end-to-end latency, we rerank only the top 96 documents per query and truncate reranker input to a maximum of 256 tokens.

The Covidex was built using the FastAPI Python web framework, where all incoming API requests are handled by a service that performs searching, reranking, and text highlighting. Search is performed with Pyserini (Section 2.1), and the results are then reranked with PyGaggle (Section 2.2). The frontend (which is also open source) was built with the React JavaScript library to support the use of modular, declarative components, taking advantage of its vast ecosystem.

A screenshot of our system is shown in Figure 1. Covidex provides standard search capabilities, either based on keyword queries or natural-language input. Users can click "Show more" to reveal the abstract as well as excerpts from the full text, where potentially relevant passages are highlighted. Clicking on the title brings the user to the article's source on the publisher's site. In addition, we have implemented a faceted browsing feature. From CORD-19, we were able to easily expose facets corresponding to dates, authors, journals, and sources. Navigating by year, for example, allows a user to focus on older coronavirus research (e.g., on SARS) or the latest research on COVID-19, and a combination of the journal and source facets allows a user to differentiate between preprints and the peer-reviewed literature, and between venues with different reputations.

The system is currently deployed across a small cluster of servers, each with two NVIDIA V100 GPUs, as our pipeline requires neural network inference at query time. Each server runs the complete software stack in a simple replicated setup (no partitioning). On top of this, we leverage Cloudflare as a simple load balancer, which uses a round robin scheme to dispatch requests across the different servers. The end-to-end latency for a typical query is around two seconds.

The first implementation of our system was deployed in late March, and we have been incrementally adding features since. Based on Cloudflare statistics, our site receives around two hundred unique visitors per day and the site serves more than one thousand requests each day. Of course, usage statistics were (up to several times) higher when we first launched due to publicity on social media. However, the figures cited above represent a "steady state" that has held up over the past few months, in the absence of any deliberate promotion.

4 TREC-COVID

Reliable, large-scale evaluations of text retrieval methods are a costly endeavor, typically beyond the resources of individual research groups. Fortunately, the community-wide TREC-COVID challenge sponsored by the U.S. National Institute for Standards and Technology (NIST) provided a forum for evaluating our techniques.

4.1 Evaluation Overview

The TREC-COVID challenge, which ran from mid-April to late-August 2020, provided an opportunity for researchers to study methods for quickly standing up information access systems, both in response to the current pandemic and to prepare for similar future events (Voorhees et al., 2020; Roberts et al., 2020). The challenge was open to everyone.

Both out of logistic necessity in evaluation design and because the body of scientific literature is rapidly expanding, TREC-COVID was organized into a series of "rounds" (five in total), each of which used the CORD-19 collection at a snapshot in time. For a particular round, participating teams developed systems that return results for a number of information needs, called "topics"—one example is "serological tests that detect antibodies of COVID-19". These results comprise a run or a submission. Each team could submit up to three runs per round (increased to eight in the final round). NIST then gathered and evaluated these runs using a standard pooling methodology (Voorhees, 2002).

The product of each round was a collection of relevance judgments, which are annotations by domain experts about the relevance of documents with respect to topics. On average, there were around 300 judgments (both positive and negative) *per topic* from each round. These relevance judgments were used to evaluate the effectiveness of systems (populating a leaderboard) and could also be used to train machine-learning models in future rounds. Runs that took advantage of these relevance judgments were known as "feedback" runs, in contrast to "automatic" runs that did not. A third category, "manual" runs, could involve human input, but we did not submit any such runs.

The TREC-COVID challenge spanned a total of five rounds. Each round contained a number of topics that were persistent (i.e., carried over from previous rounds) as well as new topics. To avoid retrieving duplicate documents, the evaluation adopted a residual collection methodology, where judged documents (either relevant or not) from previous rounds were automatically removed from consideration. Thus, for each topic, future rounds only evaluated documents that had not been examined before (either newly published articles or articles that had never been retrieved). It is worth emphasizing that due to the evaluation methodology, scores across rounds *are not* comparable.

The official evaluation metric was nDCG, at rank cutoff 10 for the first three rounds, increased to 20 for rounds 4 and 5. NIST also reported a few other metrics, including precision at a fixed ranked cutoff and average precision (AP) to the standard rank depth of 1000.

4.2 Results

A selection of results from TREC-COVID is shown in Table 1, where we report standard metrics computed by NIST. We submitted runs under team "covidex" (for neural models) and team "anserini" (for our bag-of-words baselines). In our narrative below, when we refer to rank positions (e.g., the second-best run), we disregard multiple runs from the same team. For complete details of all results, we refer readers to the official NIST site[5] or a mirror of the results in an easily comparable format that we have compiled.[6]

In **Round 1**, there were 143 runs from 56 teams. Our best run `T5R1` (1c) used BM25 for first-stage retrieval using the paragraph index followed by our monoT5-3B reranker, trained on MS MARCO (as described in Section 2.2). The best automatic neural run was `run2` (1b) from team GUIR_S2 (MacAvaney et al., 2020), which was built on Anserini. This run placed second behind the best automatic run, `sabir.meta.docs` (1a), which interestingly was based on the vector-space model.

While we did make meaningful infrastructure contributions (e.g., Anserini provided the keyword search results that fed the neural ranking models of team GUIR_S2), our own run `T5R1` (1c) was substantially behind the top-scoring runs. A post-hoc experiment with round 1 relevance judgments showed that using the paragraph index did not turn out to be the best choice: simply replacing with the abstract index (but retaining the monoT5-3B reranker) improved nDCG@10 from 0.5223 to 0.5702.[7]

We learned two important lessons from the results of round 1:

1. The effectiveness of simple rank fusion techniques that can exploit diverse ranking signals by combining multiple ranked lists. Many teams adopted such techniques (including the top-scoring run), which proved both robust and effective. This is not a new observation in information retrieval, but is once again affirmed by TREC-COVID.

2. The importance of building the "right" query representations for keyword search. Each TREC-COVID topic contains three fields: query, question, and narrative. The query field describes the information need using a few keywords, similar to what a user would type into a web search engine. The question field phrases the information need as a well-formed natural language question, and the narrative field contains additional details in a short paragraph. The query field may be missing important keywords, but the other two fields often contain too many "noisy" terms unrelated to the information need.

   Thus, it makes sense to leverage information from multiple fields in constructing keyword queries, but to do so selectively. Based on results from round 1, the following query genera-

[5]https://ir.nist.gov/covidSubmit/
[6]https://github.com/castorini/TREC-COVID

[7]Despite this finding, we suspected that there may have been evaluation artifacts at play, because our impressions from the deployed system suggested that results from the paragraph index were better. Thus, the deployed Covidex still uses paragraph indexes.

	Team	Run	Type	nDCG@10	P@5	AP
Round 1: 30 topics						
(1a)	sabir	`sabir.meta.docs`	automatic	0.6080	0.7800	0.3128
(1b)	GUIR_S2	`run2`†	automatic	0.6032	0.6867	0.2601
(1c)	covidex	`T5R1` (= monoT5)	automatic	0.5223	0.6467	0.2838
Round 2: 35 topics						
(2a)	mpiid5	`mpiid5_run3`†	manual	0.6893	0.8514	0.3380
(2b)	CMT	`SparseDenseSciBert`†	feedback	0.6772	0.7600	0.3115
(2c)	UIowaS	`UIowaS_Run3`	feedback	0.6382	0.7657	0.2845
(2d)	GUIR_S2	`GUIR_S2_run1`†	automatic	0.6251	0.7486	0.2842
(2e)	covidex	`covidex.t5` (= monoT5)	automatic	0.6250	0.7314	0.2880
(2f)	anserini	`r2.fusion2`	automatic	0.5553	0.6800	0.2725
(2g)	anserini	`r2.fusion1`	automatic	0.4827	0.6114	0.2418
Round 3: 40 topics						
(3a)	covidex	`r3.t5_lr` (= monoT5 + LR)	feedback	0.7740	0.8600	0.3333
(3b)	BioinformaticsUA	`BioInfo-run1`	feedback	0.7715	0.8650	0.3188
(3c)	SFDC	`SFDC-fus12-enc23-tf3`†	automatic	0.6867	0.7800	0.3160
(3d)	covidex	`r3.duot5` (= monoT5 + duoT5)	automatic	0.6626	0.7700	0.2676
(3e)	covidex	`r3.monot5` (= monoT5)	automatic	0.6596	0.7800	0.2635
(3f)	anserini	`r3.fusion2`	automatic	0.6100	0.7150	0.2641
(3g)	anserini	`r3.fusion1`	automatic	0.5359	0.6100	0.2293
	Team	**Run**	Type	nDCG@20	P@20	AP
Round 4: 45 topics						
(4a)	unique_ptr	`UPrrf38rrf3-r4`†	feedback	0.7843	0.8211	0.4681
(4b)	covidex	`covidex.r4.duot5.lr` (= expando + monoT5 + duoT5 + LR)	feedback	0.7745	0.7967	0.3846
(4c)	covidex	`covidex.r4.d2q.duot5` (= expando + monoT5 + duoT5)	automatic	0.7219	0.7267	0.3122
(4d)	covidex	`covidex.r4.duot5` (= monoT5 + duoT5)	automatic	0.6877	0.6922	0.3283
(4e)	uogTr	`uogTrDPH_QE_SCB1`	automatic	0.6820	0.7144	0.3457
(4f)	anserini	`r4.fusion2`	automatic	0.6089	0.6589	0.3088
(4g)	anserini	`r4.fusion1`	automatic	0.5244	0.5611	0.2666
Round 5: 50 topics						
(5a)	unique_ptr	`UPrrf93-wt-r5`†	feedback	0.8496	0.8760	0.4718
(5b)	covidex	`covidex.r5.2s.lr` (= monoT5 + duoT5 + LR)	feedback	0.8311	0.8460	0.3922
(5c)	covidex	`covidex.r5.d2q.2s.lr` (= expando + monoT5 + duoT5 + LR)	feedback	0.8304	0.8380	0.3875
(5d)	covidex	`covidex.r5.d2q.2s` (= expando + monoT5 + duoT5)	automatic	0.7539	0.7700	0.3227
(5e)	covidex	`covidex.r5.2s` (= monoT5 + duoT5)	automatic	0.7457	0.7610	0.3212
(5f)	uogTr	`uogTrDPH_QE_SB_CB`	automatic	0.7427	0.7910	0.3305
(5g)	covidex	`covidex.r5.d2q.1s` (= expando + monoT5)	automatic	0.7121	0.7320	0.3150
(5h)	anserini	`r5.fusion2`	automatic	0.6007	0.6440	0.2734
(5i)	anserini	`r5.fusion1`	automatic	0.5313	0.5840	0.2314

Table 1: Selected TREC-COVID results. Our submissions are under teams "covidex" and "anserini". All runs notated with † incorporate our infrastructure components in some way. Note that the metrics used in the first three rounds are different from those used in the final two rounds.

tion technique proved to be effective: when constructing the keyword query for a given topic, we take the non-stopwords from the query field and further expand them with terms belonging to named entities extracted from the question field using ScispaCy (Neumann et al., 2019).

We saw these two lessons as an opportunity to further contribute community infrastructure, and starting in round 2 we made two fusion runs from Anserini freely available: `fusion1` and `fusion2`. In both runs, we combined rankings from the abstract, full-text, and paragraph indexes via reciprocal rank fusion (RRF) (Cormack et al., 2009). The runs differed in their treatment of the query representation. The run `fusion1` simply took the query field from the topics as the basis for keyword search, while run `fusion2` incorporated the query generator described above to augment the query representation with key phrases. These runs were made available *before* the deadline of subsequent rounds so that other teams could use them, and indeed many took advantage of this resource.

In **Round 2**, there were 136 runs from 51 teams. Our two Anserini baseline fusion runs are shown as `r2.fusion1` (2g) and `r2.fusion2` (2f) in Table 1. Comparing these two fusion baselines, we see that our query generation approach yields a large gain in effectiveness. Ablation studies further confirmed that ranking signals from the different indexes do contribute to the overall higher effectiveness of the rank fusion runs. That is, the effectiveness of the fusion results is higher than results from any of the individual indexes.

Our `covidex.t5` (2e) run took `r2.fusion1` (2g) and `r2.fusion2` (2f), reranked both with monoT5-3B, and then combined their outputs with reciprocal rank fusion. The monoT5-3B model was fine-tuned on MS MARCO and then fine-tuned (again) on a medical subset of MS MARCO (MacAvaney et al., 2020). This run essentially tied for the best automatic run `GUIR_S2_run1` (2d), which scored just 0.0001 higher.

As additional context, Table 1 shows the best manual and feedback runs from round 2, `mpiid5_run3` (2a) and `SparseDenseSciBert` (2b), respectively, which were also the top two runs overall. These results show that manual and feedback techniques can achieve quite a bit of gain over fully automatic techniques. Both of these runs and four out of the five top teams in round 2 took advantage of the fusion baselines we provided, which demonstrated our impact not only in developing effective ranking models, but also our service to the community in providing infrastructure.

As another point of comparison, `UIowaSRun3` (2c) represented a fusion of two traditional (i.e., term-based) relevance feedback runs, and did not use any neural networks. Interestingly, its effectiveness is not very far behind `SparseDenseSciBert` (2b), the best run in the feedback category (which does take advantage of BERT). It seems that BERT-based methods for exploiting relevance judgments yielded only modest improvements, likely due to the paucity of relevance judgments, as we discussed in Section 2.2.

In **Round 3**, there were 79 runs from 31 teams. Our Anserini fusion baselines, `r3.fusion1` (3g) and `r3.fusion2` (3f), remained the same from the previous round and continued to provide strong baselines, both for our team's own submissions and other participants as well.

Our run `r3.duot5` (3d) was the first deployment of our monoT5 and duoT5 multi-stage ranking pipeline (see Section 2.2), which was a fusion of the fusion runs as the first-stage candidates, reranked by monoT5 and then duoT5. From Table 1, we see that duoT5 did indeed improve over just using monoT5, run `r3.monot5` (3e), albeit the gains were small (but we found that the duoT5 run had more unjudged documents). The run `r3.duot5` (3d) ranked second among all teams under the automatic condition, and we were about two points in nDCG@10 behind team SFDC (3c), who submitted the best run. According to Esteva et al. (2020), their general approach incorporated Anserini fusion runs, which bolsters our case that we provided valuable infrastructure for the community.

Our own feedback run `r3.t5_lr` (3a) implemented the classification-based feedback technique (see Section 2.2) with monoT5 results as the input source document (with a mixing weight of 0.5 to combine monoT5 scores with classifier scores). This was the highest-scoring run across all submissions (all categories), just a bit ahead of `BioInfo-run1` (3b).

In **Round 4**, there were 72 runs from 27 participating teams. Our Anserini fusion baselines, `r4.fusion1` (4g) and `r4.fusion2` (4f), remained exactly the same as before.

This round saw the first deployment of our doc2query document expansion technique. The run `covidex.r4.d2q.duot5` (4c) was essentially the same as `r3.duot5` from round 3, except with doc2query-enhanced indexes; in Table 1 we denote this as "expando". This run was the best automatic run submitted, about four points ahead in terms of nDCG@20 of the second-best automatic run, from team uogTr (4e). Our run `covidex.r4.duot5` (4d) adopted the same approach as `r3.duot5` from round 3 and thus provided an ablation that shows the impact of document expansion. We see from these results that doc2query contributed nearly four points in terms of nDCG@20.

Our feedback run, `covidex.r4.duot5.lr` (4b) used the same relevance classification approach as our run `r3.t5_lr` (3a) from round 3, but with doc2query-enhanced indexes. This was the second-best feedback run, about a point in nDCG@20 behind the best run from team unique_ptr (4a), which was an ensemble of more than 100 runs (Bendersky et al., 2020).

In **Round 5**, there were 126 runs from 28 participating teams. For this final round, NIST increased the number of runs that each team was allowed to submit, up to eight in total. Our Anserini fusion baselines, `r5.fusion1` (5i) and `r5.fusion2` (5h), remained exactly as before.

We used this final round as an opportunity to consolidate and refactor our codebase; no new techniques were introduced. In terms of rankings, little changed from round 4: we reported the best automatic run (5d), slightly ahead of a run from team uogTr (5f), and the second-best feedback run (5b), behind a run from team unique_ptr (5a), as shown in Table 1.

The ability to submit more runs also allowed us to conduct ablation analyses. The runs (5b) and (5c) quantified the effect of document expansion in the feedback setting. We see, quite interestingly, that document expansion has minimal effect, and in fact the run *without* document expansion achieved a slightly higher nDCG@20 (but likely just noise). Similarly, runs (5d) and (5e) quantified the effects of document expansion in the automatic setting. Contrary to what we observed in round 4, the benefits were relatively modest. Runs (5d) and (5g) quantified the effects of the pairwise reranker, with document expansion. Unlike round 3, here we see that two-stage reranking (monoT5 + duoT5) brought considerable improvement over single-stage reranking (monoT5).

For cases where we saw less improvement than expected, one possible explanation is that the topics had become "too easy", at least given the standards of relevance assessment and the evaluation metrics. This can be seen in the high absolute values of the scores. If we examine, say, precision at rank cutoff 10, most of the high-scoring runs attained scores above 0.9; that is, nine of the top ten results were relevant on average! It may be the case that the evaluation had lost discriminative power to separate runs that were all "pretty good".

5 Conclusions

Our project has three goals: build community infrastructure, advance the state of the art in neural ranking, and provide a useful application. We believe that we have succeeded in all three goals. Beyond COVID-19, the capabilities we've developed can be applied to analyzing the scientific literature more broadly.

Acknowledgments

This research was supported in part by the Canada First Research Excellence Fund, the Natural Sciences and Engineering Research Council (NSERC) of Canada, CIFAR AI & COVID-19 Catalyst Funding 2019–2020, and Microsoft AI for Good COVID-19 Grant. We'd like to thank Kyle Lo from AI2 for helpful discussions and Colin Raffel from Google for his assistance with T5.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)*, pages 265–283.

Nima Asadi and Jimmy Lin. 2013. Effectiveness/efficiency tradeoffs for candidate generation in multi-stage retrieval architectures. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2013)*, pages 997–1000, Dublin, Ireland.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv:1611.09268v3*.

Michael Bendersky, Honglei Zhuang, Ji Ma, Shuguang Han, Keith Hall, and Ryan McDonald. 2020. RRF102: Meeting the TREC-COVID challenge with a 100+ runs ensemble. *arXiv:2010.00200*.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. 2009. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009)*, pages 758–759, Boston, Massachusetts.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 985–988, Paris, France.

Andre Esteva, Anuprit Kale, Romain Paulus, Kazuma Hashimoto, Wenpeng Yin, Dragomir Radev, and Richard Socher. 2020. CO-Search: COVID-19 information retrieval with semantic search, question answering, and abstractive summarization. *arXiv:2006.09595*.

Cheolhyoung Lee, Kyunghyun Cho, and Wanmo Kang. 2020. Mixout: Effective regularization to finetune large-scale pretrained language models. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Jimmy Lin. 2009. Is searching full text more effective than searching abstracts? *BMC Bioinformatics*, 10:46.

Jimmy Lin. 2019. The simplest thing that can possibly work: pseudo-relevance feedback using text classification. *arXiv:1904.08861*.

Shichen Liu, Fei Xiao, Wenwu Ou, and Luo Si. 2017. Cascade ranking for operational e-commerce search. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2017)*, pages 1557–1565, Halifax, Nova Scotia, Canada.

Sean MacAvaney, Arman Cohan, and Nazli Goharian. 2020. SLEDGE: A simple yet effective baseline for coronavirus scientific knowledge search. *arXiv:2005.02365*.

Sean MacAvaney, Andrew Yates, Arman Cohan, and Nazli Goharian. 2019. CEDR: Contextualized embeddings for document ranking. In *Proceedings of the 42nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2019)*, pages 1101–1104, Paris, France.

Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006)*, pages 437–444, Seattle, Washington.

Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. 2019. ScispaCy: Fast and robust models for biomedical natural language processing. *arXiv:1902.07669*.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020a. Document ranking with a pretrained sequence-to-sequence model. *arXiv:2003.06713*.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020b. Document ranking with a pretrained sequence-to-sequence model. In *Findings of EMNLP*.

Rodrigo Nogueira and Jimmy Lin. 2019. From doc2query to docTTTTTquery.

Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. 2019a. Multi-stage document ranking with BERT. *arXiv:1910.14424*.

Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019b. Document expansion by query prediction. *arXiv:1904.08375*.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Jan Pedersen. 2010. Query understanding at Bing. In *Industry Track Keynote at the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2010)*, Geneva, Switzerland.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Kirk Roberts, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, Kyle Lo, Ian Soboroff, Ellen Voorhees, Lucy Lu Wang, and William R. Hersh. 2020. TREC-COVID: Rationale and structure of an information retrieval shared task for COVID-19. *Journal of the American Medical Informatics Association*.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC-3)*, pages 109–126, Gaithersburg, Maryland.

Amit Singhal, Chris Buckley, and Mandar Mitra. 1996. Pivoted document length normalization. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1996)*, pages 21–29, Zürich, Switzerland.

Ellen Voorhees. 2002. The philosophy of information retrieval evaluation. In *Evaluation of Cross-Language Information Retrieval Systems: Second Workshop of the Cross-Language Evaluation Forum, Lecture Notes in Computer Science Volume 2406*, pages 355–370.

Ellen Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: Constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1):1–12.

Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2011)*, pages 105–114, Beijing, China.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. CORD-19: The COVID-19 Open Research Dataset. *arXiv:2004.10706*.

Peilin Yang, Hui Fang, and Jimmy Lin. 2017. Anserini: enabling the use of Lucene for information retrieval research. In *Proceedings of the 40th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2017)*, pages 1253–1256, Tokyo, Japan.

Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: reproducible ranking baselines using Lucene. *Journal of Data and Information Quality*, 10(4):Article 16.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota.

Zeynep Akkalyoncu Yilmaz, Charles L. A. Clarke, and Jimmy Lin. 2020. A lightweight environment for learning experimental IR research practices. In *Proceedings of the 43rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020)*.

Zeynep Akkalyoncu Yilmaz, Wei Yang, Haotian Zhang, and Jimmy Lin. 2019. Cross-domain modeling of sentence-level evidence for document retrieval. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3481–3487, Hong Kong, China.

Ruifan Yu, Yuhao Xie, and Jimmy Lin. 2019. Simple techniques for cross-collection relevance feedback. In *Proceedings of the 41th European Conference on Information Retrieval, Part I (ECIR 2019)*, pages 397–409, Cologne, Germany.

Edwin Zhang, Nikhil Gupta, Rodrigo Nogueira, Kyunghyun Cho, and Jimmy Lin. 2020a. Rapidly deploying a neural search engine for the COVID-19 Open Research Dataset: Preliminary thoughts and lessons learned. *arXiv:2004.05125*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Revisiting few-sample BERT fine-tuning. *arXiv:2006.05987*.