A Modular Framework for Conversational Search Reproducible Experimentation*

Discussion Paper

Marco Alessio[1], Guglielmo Faggioli[1] and Nicola Ferro[1]

[1]University of Padova, Padova, Italy

Abstract

The Conversational Search (CS) paradigm facilitates users' interaction with IR systems through natural language sentences, and it is increasingly being used in various scenarios. However, the proliferation of custom conversational search systems and components makes it challenging to compare and and design new CS agents. To tackle this issue, we propose DECAF: a modular and extensible conversational search framework that enables rapid development of conversational agents. DECAF integrates all the necessary components of a modern conversational search system into a uniform interface. DECAF allows for experiments that exhibit a high degree of reproducibility, addressing the reproducibility crisis in the field. DECAF framework includes several state-of-the-art components such as query rewriting, search functions under BoW and dense paradigms, and re-ranking functions. We evaluate the DECAF on two well-known conversational collections, CAsT '19 and CAsT '20, and provide the results as baselines for future practitioners.

1. Introduction

Conversational Search (CS) [2, 3] is an emerging paradigm which is drastically innovating Information Retrieval (IR) by allowing users to issue their queries to the system in the form of a conversation. This paradigm allows for a very intuitive and seamless interaction between the human and the system since it is based on natural language sentences. Nevertheless, it also presents important challenges due to the presence of complex speech structures in the utterances, such as anaphoras, ellipses or coreferences. Therefore, the system needs to address these natural language phenomena by keeping track of the conversation state.

The advent of Neural Information Retrieval (NIR) models, fueled by Large Language Models (LLMs), has been helping to overcome some of these challenges and to foster a capillary adoption of CS in many different scenarios. Such techniques, given their complexity, were originally used mostly for re-ranking [4, 5, 6]. With their improvement both in terms of effectiveness and efficiency, we also witnessed their adoption as first-stage retrieval systems [7, 8, 9]. The plethora of CS systems that rely on these building blocks allow for addressing the user's information need in a number of different scenarios. Nevertheless, such a variety made the CS scenario fragmented from the systems' design perspective, with hundreds of ad-hoc custom

SEBD 2023: 31st Symposium on Advanced Database System, July 02–05, 2023, Galzignano Terme, Padua, Italy

*This is an extended abstract of [1]

marco.alessio@studenti.unipd.it (M. Alessio); guglielmo.faggioli@unipd.it (G. Faggioli)

implementations and variants of IR models and other components, implemented using a wide range of different languages and frameworks, often difficult to integrate together, if not impossible at all. This causes a number of challenges both in the development of CS systems and in their experimentation. Firstly, it is difficult to combine various state-of-the-art components together, since they often come from different and/or incompatible libraries and packages; in turn, this hampers the development of new and competitive approaches. Secondly, alternative implementations of a component often lead to different performance or behavior; in turn, this hampers the comparability of different end-to-end solutions, integrating alternative versions of the same components. Thirdly, it is extremely difficult, if not impossible, to conduct systematic experiments where different components are combined in all the possible ways, in a grid-like way [10], in order to break down the contribution of different components and analyse their interactions [11]. Finally, it hinders the reproducibility of experiments, exacerbating the already prevalent reproducibility crisis in current research [12, 13, 14, 15].

To address these issues, we propose **DECAF** – *mo**D**ular and **E**xtensible **C**onversational sea**A**rch **F**ramework*, which is explicitly designed to allow for fast prototyping and development of CS systems and to enable their systematic evaluation under the traditional Cranfield paradigm. The framework is designed to allow the integration of all the components that characterize a modern CS system, including query rewriting, search – both under the Bag-of-Words (BoW) and dense paradigms – and re-ranking. While written primarily in Java to ensure compatibility with traditional IR libraries, such as Lucene[1], Terrier [16], and Anserini [17], it also supports modern Artificial Intelligence (AI), Machine Learning (ML), and LLM-based techniques thanks to the seamless integration of Python scripts. The framework already includes multiple state-of-the-art (sota) components for query rewriting, traditional BoW similarity functions, NIR approaches – both sparse and dense – and re-ranking functions. It also allows for the evaluation on reference collections in the area, such as TREC CAsT 2019 and TREC CAsT 2020. The components implemented in DECAF act as state-of-the-art baselines for future experiments. They also provide a template for integrating and designing new components, to extend DECAF itself. To show the capabilities of this framework, we demonstrate its application to the TREC CAsT 2019 and TREC CAsT 2020 collections, reporting the results that different pipelines can achieve. Our main contributions are the following:

- Design and develop the DECAF modular and easily extensible framework, which allows us to seamlessly integrate CS components and instantiate CS pipelines.
- Provide a series of state-of-the-art components that can be used to implement a CS pipeline.
- Evaluate several CS pipelines, created using DECAF, on two well-known and widely adopted conversational collections, namely TREC CAsT 2019 and TREC CAsT 2020.

2. DECAF Architecture

In this section, we provide the overview of DECAF, focusing on, the principal modules of its architecture, components implemented, and software requirements.

[1] https://lucene.apache.org/

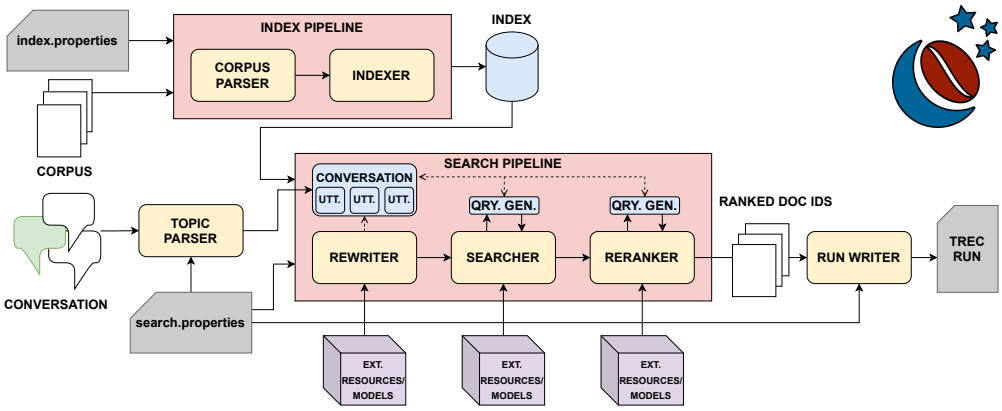

Figure 1: DECAF Architecture.

2.1. Main Modules

DECAF relies on a modular architecture for index and search pipelines. In practice, to foster extensibility, as well as standardization, each module of DECAF is defined as a Java interface. As illustrated in Figure 1, the index pipeline revolves around two main modules: the *Corpus Parser* and the *Indexer*. The former processes the corpus into a stream of documents, which is consumed by the latter to index them. For the search pipeline, we adopt a multi-stage architecture which employs the modules that are the most common for CS systems.

Figure 1 shows the structure of the search pipeline:

- The *Topics Parser* reads in input a file – e.g., written in TREC format – and provides parsed conversations and utterances to be processed by the rest of the framework.
- The *Rewriter* modifies the text of the utterances by performing pronoun disambiguation and adding contextual information extracted from previous utterances in the conversation.
- The *Searcher* takes the (possibly rewritten) utterance text as input, generates the query and retrieves a set of candidate documents to answer the provided question.
- The ranked list of documents generated as output by the Searcher is consumed by the *Reranker*. This module is designed to apply complex and resource-consuming re-ranking operations upon the Searcher output, boosting the performance of the CS pipeline.

Furthermore, we exploit two additional utility models: the *Query Generator* and the *Run Writer*. Both the searcher and the re-ranker modules exploit the *Query Generator* to obtain a representation of the user utterance — possibly by combining it with previous ones — that is directly used at retrieval time. Finally, the *Run Writer* is a utility module meant to write the run on a file, so that it can be further used or evaluated.

2.2. Components Implemented

Requirements The core of DECAF has been developed in Java, with the integration of Python for machine-learning-oriented functionalities. It requires Java Development Kit (JDK) 11 and

Python 3.8 for execution. The framework is built upon Lucene 8.8.1.

Every component is expected to implement the Java interface of the specific module, which defines one or more methods specific to the performed job. The components implemented with DECAF and described in the remainder of this section can be used by practitioners both as baselines as well as templates to guide practitioners in extending DECAF. Every component of the framework has some configurable parameters, which must be passed through constructor arguments. We also implemented a configuration system, based on *.properties* files, that allows the user to specify them in a user-friendly manner.

Corpus Parser We implement three components that perform corpus parsing. The first processes the passages contained within MS-MARCO version 1[2] dataset. Another parser addresses the paragraph corpus of TREC CAR v2.0 [3]. The third parser processes any corpus based on tab-separated files using the "`ID <tab> Text <newline>`" format. Furthermore, it might be necessary to index documents from multiple sources with different formats at once. The last corpus parser component eases this by allowing to instantiate multiple parsers that are used to parse different corpora.

Indexer The framework comes with three distinct indexer components a BoW indexer, a SPLADE indexer, and a Dense indexer. The BoW indexer is a wrapper around Lucene indexing operations. The BoW indexer component has been extended for the SPLADE indexer which is specific to the homonyms neural retrieval model. It replaces the standard tokenization and analysis pipeline performed by Lucene with the SPLADE model inference. Finally, the dense indexer component is specific for dense retrieval models. It is built on top of two well-known libraries: Transformers [18] and Facebook AI Similarity Search (FAISS) [19].

Conversation and Utterance Components To provide a unified interface to the data, we define two components, called *Utterance* and *Conversation*. The former is a data structure that provides unified access to the utterance and subsequent transformations operated by different components. The latter provides utilities to access the data and groups together utterances belonging to the same conversation. At runtime, each module will extract the needed data (e.g., the textual content of the utterance, its rewritten version, or the ranked list associated) from the Utterance component, passing through the Conversation interface. Upon completion of the required operations, each module will save the computed results for a specific utterance (e.g., a new rewriting of the utterance, the re-ranked run), within the Utterance component, so that the next module can access it.

The vast majority of implemented components take only two parameters: the Conversation object containing the data regarding the specific conversation at hand, and the ID of the current utterance on which we have to operate (e.g., we want to rewrite or we are retrieving the documents for). This approach ensures great flexibility in the design of the components since it is possible to access the entire data structure for the whole conversation – in particular to previously issued utterances and retrieved responses. Furthermore, it allows for easily expanding

[2] https://msmarco.blob.core.windows.net/msmarcoranking/collection.tar.gz
[3] https://trec-car.cs.unh.edu/datareleases/v2.0/paragraphCorpus.v2.0.tar.xz

the framework, since each component can behave as a black-box building block operating only on the Conversation and Utterance objects.

Topics Parser DECAF provides five topic parsers designed explicitly to handle TREC CAsT 2019 and TREC CAsT 2020 evaluation topics. More in detail, for each collection, DECAF has a parser for each type of utterance – i.e., either manual or automatic utterances. This component takes in input the topics file and generates in output a stream of Conversation objects. Each of them is further split into the individual Utterances that compose it.

Rewriter The Rewriter module expands the original text of the utterance into the rewritten text and stores it within the specific Utterance object. In DECAF, we provide two sota rewriting approaches using off-the-shelf resources, either employing coreference resolution libraries or pre-trained T5 models. In implementation terms, two library-based components carry out coreference resolution. In particular, we implement one component based on AllenNLP framework [20] and one using Fastcoref. Fastcoref is a coreference resolution utility based on the LingMess [21] architecture. In Table 1, we dub this approach CR. For the second approach, we employ a T5 model [22]. The particular instance of T5 used in the experimental part is publicly available and pre-trained specifically for conversational search question rewriting[4]. To maintain the modularity of the framework, we also implement a rewriter which corresponds to the "identity" operation and returns the original text unchanged. It should be used in all cases where the utterances have been rewritten externally from the framework or if the practitioner does not wish to carry out any form of rewriting.

Query Generator The whole conversation is considered as input for components implemented within this module, producing as output a representation of the utterance that embeds the context. Within DECAF, we provide three different query generator components. The FLC query generator takes in input the utterances for a given conversation and outputs a weighted sum of the rewritten text of the First, Last and Current (FLC) utterances. The rationale is that the first utterance often gives the general topic of the conversation, while the previous one is the most likely to be referenced again by the current utterance. The output of this component is meant to synergize with the rewriter's effort to bring contextual information into the current query, especially useful when the quality of its output is less than ideal. A second query generator provided within DECAF considers the concatenation of the rewritten text for all previous utterances of the current conversation. The final query generator implemented (dubbed C in Table 1) considers only the – possibly rewritten – text of the current utterance, without taking into account any of the previous ones. To operate, the query generator access the Utterance object referring to the current utterance, and possibly to the Utterance objects defined for previous utterances.

Searcher The searchers are specular to the indexers used in the indexing phase, with three different searcher components, one for BoW Lucene-based similarity functions, one for SPLADE and one for dense models.

[4] https://huggingface.co/castorini/t5-base-canard

The first Searcher component, the BoW one, by being based on Lucene, can be instantiated with any of the classical BoW models already implemented in it, such as BM25 [23], Language Model (LM) [24] or the Vector Space Model [25]. For example, in the experimental benchmarking reported in Section 3, we exploit BM25. BM25 [23] is BoW IR model that ranks documents based on the occurrence and frequency of terms. The second similarity function that we take into consideration is SPLADE. SPLADE [9, 26] is a sparse NIR model that learns sparse representation for both queries and documents via the BERT MLM head and sparse regularization.We separate it from the previous component, even though they are both BoW because SPLADE requires computing the BoW sparse representation of the query. In our experimental analysis, we use a publicly available set of weigths[5]. Finally, we implement a component for dense retrieval that can be used with FAISS indexes. In particular, we instantiate it with BERT. We exploit a publicly available BERT instance fine-tuned specifically for IR[6]. Components implementing this module invoke the Query Generator sub-component that provides them with a searcher-specific query representation directly used for retrieval. Upon retrieval completion, components within the Searcher module must save the top-k retrieved document IDs within the Utterance component.

Reranker This module must set the final ranked list of documents in the Utterance object. This final ranking is the output of the whole search pipeline for the current utterance.

Components implemented within the reranker module consider the documents included in the ranking list produced by the Searcher and generate a new relevance score for each of them. At the current time, DECAF comes with a reranker component: Transformers. The Transformers reranker employs the Transformers (Hugging Face) [18] library to apply machine-learning models, such as BERT, to the text of both the rewritten query and of the documents retrieved by the Searcher, then evaluate the similarity between them. Notice that the specific transformers model used can be chosen at runtime, by specifying it in the properties file. We experiment with BERT, since several works already observed the effectiveness of BERT for the re-ranking task in the CS and IR domains [27, 28, 29]. Finally, we assume that a user might not be interested in re-ranking documents. In that case, we included the "identity" reranker which returns the ranked list of documents generated by the Searcher.

Run Writer This final module can be instantiated into two modalities. The first component – dubbed Trec Eval – produces a run using the standard TREC eval format. In particular, it saves inside the *runs* sub-folder of the framework a tab-separated file with six columns: the query id, the Q0 placeholder, the document id and ranking, the retrieval score, and the user-specified id of the run. Secondly, to ease the debugging, the "debug" component saves on file both ranked lists produced by the Searcher and Reranker together with a file containing the top-k documents retrieved by the system to allow for manual inspection and precise failure analysis. DECAF is available as open source under the *Creative Commons Attribution-ShareAlike 4.0 International License* at the following address: https://github.com/alemarco96/DECAF.

[5] https://huggingface.co/naver/efficient-splade-V-large-query
[6] https://huggingface.co/sebastian-hofstaetter/distilbert-dot-tas_b-b256-msmarco

Table 1
List of all experiments conducted on TREC CAsT 2019 benchmark. The evaluation measures reported are R@100, MRR and nDCG with cutoffs 3 and 10.

	Topics	Rew.	Searcher	Rer.	Rec.	MRR	nDCG@3	nDCG@10
1	Auto.	—	BM25 c	—	19.9	32.0	14.2	14.4
2		CR	BM25 c	—	35.3	51.5	25.9	25.4
3		T5	BM25 c	—	42.8	64.0	33.9	32.1
4		—	BM25 FLC	—	23.9	41.0	19.1	18.4
5		T5	BM25 FLC	—	45.0	65.6	34.4	32.5
6		—	BM25 c	BERT	19.9	48.0	28.4	24.9
7		T5	BM25 c	BERT	42.8	79.3	50.4	45.2
8		—	BM25 FLC	BERT	23.9	51.7	30.9	27.5
9		T5	BM25 FLC	BERT	45.0	79.4	50.3	46.0
10		T5	BERT c	—	43.2	52.3	30.4	33.1
11		—	SPLADE c	—	24.8	44.8	27.5	26.7
12		T5	SPLADE c	—	**51.5**	**79.9**	**52.3**	**50.1**
Baseline Automatic at TREC CAsT 2019					21.4	31.8	14.8	15.9
Best Automatic Run at TREC CAsT 2019					41.2	71.1	42.4	40.6
13	Man.	—	BM25 c	—	47.8	66.7	35.4	34.5
14		—	BM25 c	BERT	47.8	82.5	54.4	48.2
15		—	BERT c	—	46.4	54.3	32.8	35.5
16		—	SPLADE c	—	54.9	84.3	56.6	53.5
Best Manual Run at TREC CAsT 2019					**56.7**	**88.4**	**59.3**	**60.6**

3. Experimental Results

The framework has been tested on TREC CAsT 2019 and TREC CAsT 2020 collections[7]. We report four used measures: Recall (R)@100, Mean Reciprocal Rank (MRR), normalized Discounted Cumulative Gain (nDCG)@3, and nDCG@10, computed using the `trec_eval`[8] tool. Following the official evaluation settings on TREC CAsT 2020 dataset, we consider documents as relevant if their relevance score is ≥ 2 [30].

Table 1 shows the experimental results on TREC CAsT 2019. The initial experiment (#1) evaluates the performance of the BM25 model on automatic utterances: the system performs poorly due to the lack of contextual information. Results are in line with those observed by [31] with respect to the baseline used for TREC CAsT 2019 (LMs). Experiments #2 and #3 add the Rewriter component to the pipeline, which brings forward implied substantives into the text of the utterance, resulting in improved performance. Run #3 employs T5 and achieves double the performance w.r.t the automatic baseline across all measures. The FLC variant (used in Experiments #4 and #5) combines the first, previous and current utterances' text. This simple heuristic is beneficial, particularly in cases where the rewriter fails to correctly expand and disambiguate pronouns with the relative entity. Again, results are consistent with the

[7]Due to space reasons, we report results only for TREC CAsT 2019. See [1] for TREC CAsT 2020 results.
[8]https://github.com/usnistgov/trec_eval

observations made in the TREC CAsT 2019 overview [31]. Experiments #6 to #9 demonstrate that performing an additional re-ranking step using a BERT model can further improve performance for all experimental settings. A comparison between experiment #1 and #6 shows that, despite the absence of any form of rewriting, the performance is significantly improved by using BERT as reranker, thus alleviating the issues related to the "context representation" introduced by the CS setting. The performance of our implementation of the BERT-based re-ranking strategy is similar to the average performance of systems submitted at TREC CAsT 2019 [31]. If the same BERT model is used for first-stage dense retrieval (Experiment #10), it achieves performance similar to a standard lexical model such as BM25 (experiment #3).Experiments #11 and #12 evaluate SPLADE as a first-stage retrieval method. The particular SPLADE instance employed has been fine-tuned for passage retrieval, utilizing two distinct models for documents and queries. It performs document expansion by adding a large number of terms not found in the original text while producing sparser representations for queries. Test #12 shows that SPLADE achieves the best results, outperforming BM25 with BERT (experiment #9) by 14.4% in terms of recall, 4.0% for nDCG@3, and 8.9% for nDCG@10 when the T5-based rewriter is used. It also surpasses all original TREC CAsT 2019 automatic submissions, obtaining an improvement against the Best Automatic (BA) of respectively 25.0%, 12.4%, 23.3%, and 23.4% for the four measures considered. We now focus on the second part of Table 1, where the manually rewritten utterances are used. Notice that, the performance observed on this kind of utterance represents an upper bound on the performance that can be achieved by retrieval systems. In fact, in a real case scenario, such utterances would not be available. The performance differences between automatic and manual utterances are mostly independent of the retrieval model utilized, with average differences of 8.6% for recall, 4.5% for MRR, 8.0% for nDCG@3 and 6.9% for nDCG@10. These experiments show that T5-based query rewriting methods are very effective on the TREC CAsT 2019 dataset. The best performance is achieved again using the SPLADE model, similar to the results observed for the automatic runs. When comparing our best manual run to the Best Manual (BM) run among the original submissions, we observe slightly lower scores, especially for the nDCG@10 measure with a difference of 13.3%.

4. Conclusions

In this work, we have presented DECAF, a novel resource for conducting experiments within the Conversational Information Seeking (CIS) scenario. This work is motivated by the constantly growing plethora of heterogeneous CS systems that have been recently devised thanks to the advent of LLMs. DECAF has been designed to favour comparability between systems, fast prototyping and reproducibility, and in turn, alleviate the current reproducibility crisis. Therefore, DECAF has been designed around three key features: modularity, expandability and reproducibility. DECAF comes with a set of state-of-the-art components already implemented, including query rewriting, searching and re-ranking. The framework is also flexible enough to integrate additional components without much effort. We have evaluated several CS pipelines instantiated through DECAF on two well-known collections, TREC CAsT 2019 and TREC CAsT 2020 . Future work will concern the extension of DECAF with new components as well as support for the mixed-initiative task.

References

[1] M. Alessio, G. Faggioli, N. Ferro, DECAF: a Modular and Extensible Conversational Search Framework, in: SIGIR '23: The 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, Taipei, Taiwan, July 23-27, 2023, ACM, 2021.

[2] H. Zamani, J. R. Trippas, J. Dalton, F. Radlinski, Conversational Information Seeking. An Introduction to Conversational Search, Recommendation, and Question Answering, arXiv.org, Information Retrieval (cs.IR) arXiv:2201.08808 (2022).

[3] A. Anand, L. Cavedon, M. Hagen, H. Joho, M. Sanderson, B. Stein, Conversational search - A report from dagstuhl seminar 19461, CoRR abs/2005.08658 (2020). URL: https://arxiv.org/abs/2005.08658. arXiv:2005.08658.

[4] K. Hui, A. Yates, K. Berberich, G. de Melo, PACRR: A position-aware neural IR model for relevance matching, in: M. Palmer, R. Hwa, S. Riedel (Eds.), Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017, Association for Computational Linguistics, 2017, pp. 1049–1058. URL: https://doi.org/10.18653/v1/d17-1110. doi:10.18653/v1/d17-1110.

[5] C. Xiong, Z. Dai, J. Callan, Z. Liu, R. Power, End-to-end neural ad-hoc ranking with kernel pooling, in: N. Kando, T. Sakai, H. Joho, H. Li, A. P. de Vries, R. W. White (Eds.), Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017, ACM, 2017, pp. 55–64. URL: https://doi.org/10.1145/3077136.3080809. doi:10.1145/3077136.3080809.

[6] Z. Yang, Q. Lan, J. Guo, Y. Fan, X. Zhu, Y. Lan, Y. Wang, X. Cheng, A deep top-k relevance matching model for ad-hoc retrieval, in: S. Zhang, T. Liu, X. Li, J. Guo, C. Li (Eds.), Information Retrieval - 24th China Conference, CCIR 2018, Guilin, China, September 27-29, 2018, Proceedings, volume 11168 of *Lecture Notes in Computer Science*, Springer, 2018, pp. 16–27. URL: https://doi.org/10.1007/978-3-030-01012-6_2. doi:10.1007/978-3-030-01012-6_2.

[7] L. Xiong, C. Xiong, Y. Li, K. Tang, J. Liu, P. N. Bennett, J. Ahmed, A. Overwijk, Approximate nearest neighbor negative contrastive learning for dense text retrieval, in: 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, OpenReview.net, 2021. URL: https://openreview.net/forum?id=zeFrfgyZln.

[8] J. Zhan, J. Mao, Y. Liu, J. Guo, M. Zhang, S. Ma, Optimizing dense retrieval model training with hard negatives, in: F. Diaz, C. Shah, T. Suel, P. Castells, R. Jones, T. Sakai (Eds.), SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021, ACM, 2021, pp. 1503–1512. URL: https://doi.org/10.1145/3404835.3462880. doi:10.1145/3404835.3462880.

[9] T. Formal, B. Piwowarski, S. Clinchant, SPLADE: sparse lexical and expansion model for first stage ranking, CoRR abs/2107.05720 (2021). URL: https://arxiv.org/abs/2107.05720. arXiv:2107.05720.

[10] N. Ferro, D. Harman, CLEF 2009: Grid@clef pilot track overview, in: C. Peters, G. M. D. Nunzio, M. Kurimo, T. Mandl, D. Mostefa, A. Peñas, G. Roda (Eds.), Multilingual Information Access Evaluation I. Text Retrieval Experiments, 10th Workshop of the Cross-Language Evaluation Forum, CLEF 2009, Corfu, Greece, September 30 - October 2, 2009, Revised Selected Papers, volume 6241 of *Lecture Notes in Computer*

Science, Springer, 2009, pp. 552–565. URL: https://doi.org/10.1007/978-3-642-15754-7_68. doi:10.1007/978-3-642-15754-7_68.

[11] N. Ferro, G. Silvello, A general linear mixed models approach to study system component effects, in: R. Perego, F. Sebastiani, J. A. Aslam, I. Ruthven, J. Zobel (Eds.), Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016, ACM, 2016, pp. 25–34. URL: https://doi.org/10.1145/2911451.2911530. doi:10.1145/2911451.2911530.

[12] W. Yang, K. Lu, P. Yang, J. Lin, Critically examining the "neural hype": Weak baselines and the additivity of effectiveness gains from neural ranking models, in: B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, F. Scholer (Eds.), Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019, ACM, 2019, pp. 1129–1132. URL: https://doi.org/10.1145/3331184.3331340. doi:10.1145/3331184.3331340.

[13] M. F. Dacrema, P. Cremonesi, D. Jannach, Are we really making much progress? A worrying analysis of recent neural recommendation approaches, in: T. Bogers, A. Said, P. Brusilovsky, D. Tikk (Eds.), Proceedings of the 13th ACM Conference on Recommender Systems, RecSys 2019, Copenhagen, Denmark, September 16-20, 2019, ACM, 2019, pp. 101–109. URL: https://doi.org/10.1145/3298689.3347058. doi:10.1145/3298689.3347058.

[14] N. Ferro, Reproducibility challenges in information retrieval evaluation, ACM J. Data Inf. Qual. 8 (2017) 8:1–8:4. URL: https://doi.org/10.1145/3020206. doi:10.1145/3020206.

[15] S. Kharazmi, F. Scholer, D. Vallet, M. Sanderson, Examining additivity and weak baselines, ACM Trans. Inf. Syst. 34 (2016) 23:1–23:18. URL: https://doi.org/10.1145/2882782. doi:10.1145/2882782.

[16] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, D. Johnson, Terrier information retrieval platform, in: D. E. Losada, J. M. Fernández-Luna (Eds.), Advances in Information Retrieval, 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005, Proceedings, volume 3408 of *Lecture Notes in Computer Science*, Springer, 2005, pp. 517–519. URL: https://doi.org/10.1007/978-3-540-31865-1_37. doi:10.1007/978-3-540-31865-1_37.

[17] J. Lin, M. Crane, A. Trotman, J. Callan, I. Chattopadhyaya, J. Foley, G. Ingersoll, C. Macdonald, S. Vigna, Toward reproducible baselines: The open-source IR reproducibility challenge, in: N. Ferro, F. Crestani, M. Moens, J. Mothe, F. Silvestri, G. M. D. Nunzio, C. Hauff, G. Silvello (Eds.), Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings, volume 9626 of *Lecture Notes in Computer Science*, Springer, 2016, pp. 408–420. URL: https://doi.org/10.1007/978-3-319-30671-1_30. doi:10.1007/978-3-319-30671-1_30.

[18] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, A. M. Rush, Transformers: State-of-the-art natural language processing, in: Q. Liu, D. Schlangen (Eds.), Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, EMNLP 2020 - Demos, Online, November 16-20, 2020, Association for Computational Linguistics, 2020, pp. 38–45. URL: https://doi.org/10.18653/v1/2020.emnlp-demos.6. doi:10.18653/v1/2020.emnlp-demos.6.

[19] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with gpus, IEEE Trans. Big Data 7 (2021) 535–547. URL: https://doi.org/10.1109/TBDATA.2019.2921572. doi:10.1109/TBDATA.2019.2921572.

[20] M. Gardner, J. Grus, M. Neumann, O. Tafjord, P. Dasigi, N. F. Liu, M. E. Peters, M. Schmitz, L. Zettlemoyer, Allennlp: A deep semantic natural language processing platform, CoRR abs/1803.07640 (2018). URL: http://arxiv.org/abs/1803.07640. arXiv:1803.07640.

[21] S. Otmazgin, A. Cattan, Y. Goldberg, Lingmess: Linguistically informed multi expert scorers for coreference resolution, CoRR abs/2205.12644 (2022). URL: https://doi.org/10.48550/arXiv.2205.12644. doi:10.48550/arXiv.2205.12644. arXiv:2205.12644.

[22] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, P. J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (2020) 140:1–140:67. URL: http://jmlr.org/papers/v21/20-074.html.

[23] S. E. Robertson, H. Zaragoza, The probabilistic relevance framework: BM25 and beyond, Found. Trends Inf. Retr. 3 (2009) 333–389. URL: https://doi.org/10.1561/1500000019. doi:10.1561/1500000019.

[24] C. X. Zhai, Statistical language models for information retrieval: A critical review, Found. Trends Inf. Retr. 2 (2008) 137–213. URL: https://doi.org/10.1561/1500000008. doi:10.1561/1500000008.

[25] G. Salton, C. Buckley, Term-weighting approaches in automatic text retrieval, Inf. Process. Manag. 24 (1988) 513–523.

[26] T. Formal, C. Lassance, B. Piwowarski, S. Clinchant, SPLADE v2: Sparse lexical and expansion model for information retrieval, CoRR abs/2109.10086 (2021). URL: https://arxiv.org/abs/2109.10086. arXiv:2109.10086.

[27] I. Mele, C. I. Muntean, F. M. Nardini, R. Perego, N. Tonellotto, O. Frieder, Topic propagation in conversational search, in: J. X. Huang, Y. Chang, X. Cheng, J. Kamps, V. Murdock, J. Wen, Y. Liu (Eds.), Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020, ACM, 2020, pp. 2057–2060. URL: https://doi.org/10.1145/3397271.3401268. doi:10.1145/3397271.3401268.

[28] R. F. Nogueira, K. Cho, Passage re-ranking with BERT, CoRR abs/1901.04085 (2019). URL: http://arxiv.org/abs/1901.04085. arXiv:1901.04085.

[29] S. MacAvaney, A. Yates, A. Cohan, N. Goharian, CEDR: contextualized embeddings for document ranking, in: B. Piwowarski, M. Chevalier, É. Gaussier, Y. Maarek, J. Nie, F. Scholer (Eds.), Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2019, Paris, France, July 21-25, 2019, ACM, 2019, pp. 1101–1104. URL: https://doi.org/10.1145/3331184.3331317. doi:10.1145/3331184.3331317.

[30] J. Dalton, C. Xiong, J. Callan, Cast 2020: The conversational assistance track overview, in: E. M. Voorhees, A. Ellis (Eds.), Proceedings of the Twenty-Ninth Text REtrieval Conference, TREC 2020, Virtual Event [Gaithersburg, Maryland, USA], November 16-20, 2020, volume 1266 of *NIST Special Publication*, National Institute of Standards and Technology (NIST), 2020. URL: https://trec.nist.gov/pubs/trec29/papers/OVERVIEW.C.pdf.

[31] J. Dalton, C. Xiong, J. Callan, TREC cast 2019: The conversational assistance track overview, CoRR abs/2003.13624 (2020). URL: https://arxiv.org/abs/2003.13624. arXiv:2003.13624.