Chatty Goose: A Python Framework for Conversational Search

Edwin Zhang, Sheng-Chieh Lin, Jheng-Hong Yang,
Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin

David R. Cheriton School of Computer Science
University of Waterloo, Ontario, Canada

ABSTRACT

Chatty Goose is an open-source Python conversational search framework that provides strong, reproducible reranking pipelines built on recent advances in neural models. The framework comprises extensible modular components that integrate with popular libraries such as Transformers by HuggingFace and ParlAI by Facebook. Our aim is to lower the barrier of entry for research in conversational search by providing reproducible baselines that researchers can build on top of. We provide an overview of the framework and demonstrate how to instantiate a new system from scratch. Chatty Goose incorporates improvements to components that we introduced in the TREC 2019 Conversational Assistance Track (CAsT), where our submission represented the top-performing system. Using our framework, a comparable run can be reproduced with just a few lines of code.

CCS CONCEPTS

• **Information systems → Users and interactive retrieval**.

KEYWORDS

Multi-Stage Ranking; Query Reformulation

ACM Reference Format:
Edwin Zhang, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. Chatty Goose: A Python Framework for Conversational Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3462782

1 INTRODUCTION

In recent years, conversational search has attracted the attention of the IR and NLP communities, particularly due to the growing prominence of virtual assistants in consumer devices such as Alexa and Siri. To facilitate this emerging area of research, the TREC Conversational Assistance Track (CAsT) [5] began in 2019 with the goal of advancing work on conversational search systems. To evaluate the effectiveness of such systems, CAsT defines a passage retrieval task and provides reusable collections that include carefully constructed conversational sessions and high-quality judgments. There is substantial interest in the related task of conversational question answering—one aspect of conversational search—from the NLP community. Substantial progress has been made in recent years, driven by datasets such as CoQA [15] and QuAC [3].

However, the complexity of building a conversational search system from scratch can be intimidating to new researchers who would like to join this exciting and rapidly expanding field. Compared to *ad hoc* retrieval, an end-to-end conversational search system requires more capabilities, for example, the need for the system to handle anaphora and other language phenomena, to track sub-topic structure and dialogue state, etc. In fact, the conceptual model proposed by Penha et al. [13] defines fifteen tasks for a conversational search system, drawing from research in IR, NLP, and beyond.

Despite much work that has already been published on conversational search, an open-source, easy-to-use, flexible, and reproducible baseline system still does not exist. Since research today is driven by considerations of novelty, rapid iteration is often valued over reproducibility. Given the multitude of components that are required to construct an end-to-end system, this prioritization makes it difficult for researchers to build on each other's work. We believe that a stable and reproducible system based on mature techniques would provide a strong foundation for incorporating and evaluating advances in conversational search. Furthermore, as researchers and practitioners often need to quickly adapt their systems to different scenarios, the flexibility of a conversational search framework is also important. Finally, since the ultimate goal of a conversational search system is to engage in mixed-initiative dialogues, providing an interactive interface is a necessary component for the entire end-to-end experience. Such an interface would allow a system to be monitored and improved through direct user feedback.

To facilitate research in conversational search, we present Chatty Goose: an open-source Python conversational search framework that provides strong, reproducible reranking pipelines built on recent advances in neural models. Specifically, our framework allows researchers and practitioners to easily build and evaluate working multi-stage conversational passage retrieval systems, such as those designed for the Conversational Assistance Track (CAsT) Track at TREC. In fact, our framework incorporates many components that we introduced in the TREC 2019 CAsT evaluation, where our team submitted the top-performing run. Using Chatty Goose, a system that achieves a comparable level of effectiveness can be "wired together" with just a few lines of code.

Our framework takes advantage of popular open-source libraries such as Transformers by HuggingFace [20] and ParlAI by Facebook [11], and builds on additional components developed by our research group, including the Pyserini IR toolkit [8] and the PyGaggle library for text ranking. A conversational search system built using our framework can be easily evaluated on standard

SIGIR '21, July 11–15, 2021, Virtual Event, Canada.

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462782

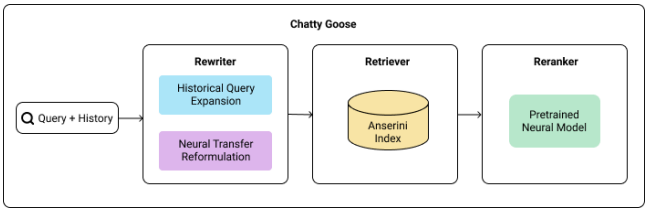

Figure 1: Architecture of the conversational passage retrieval pipeline for Chatty Goose.

benchmarks, using resources provided by TREC CAsT. We believe that Chatty Goose is a valuable contribution to the advancement of research in conversational search, as it lowers the barrier of entry and provides a solid starting point for new researchers and practitioners. At a high level, our framework is defined by the following characteristics:

- *Reproducibility.* Chatty Goose provides all components necessary to reproduce our top-performing submission to the TREC 2019 CAsT evaluation (see experimental results in Table 1). The technical innovations behind our work are described in Lin et al. [9]. Leveraging the Pyserini IR toolkit [8], we provide the infrastructure to reproduce ranking experiments using the same indexes and similar models as our submission.
- *Flexibility.* We define a canonical pipeline architecture for conversational search. While a reference implementation using our own components from the TREC 2019 CAsT evaluation can be easily instantiated, our framework provides the appropriate abstractions that allow researchers to plug in custom implementations. This lets researchers focus their efforts on specific components (e.g., query rewriting) without needing to build all components in an end-to-end system from scratch. Chatty Goose specifically supports models from the popular HuggingFace Transformers library [20], providing easy integration with modern research on neural techniques.
- *Community.* We integrate Chatty Goose with Facebook's ParlAI framework [11], which allows our conversational search system to directly interact with users through existing platforms such as Facebook Messenger, as well as access to a panoply of additional capabilities. One benefit of this integration is that it enables researchers and practitioners to collect user feedback through Amazon Mechanical Turk. In this way, Chatty Goose contrasts with other efforts such as Macaw [24] that attempt to build independent frameworks.

2 FRAMEWORK OVERVIEW

Chatty Goose is a framework designed for conversational search via a standard multi-stage ranking pipeline [1, 10, 19]. The framework provides an easy-to-use interface for building novel multi-stage pipelines and evaluating them against reproducible baselines, to reduce pain points in the research cycle for conversational search. Users can take advantage of our reference implementations or easily integrate their own components, and validate effectiveness using collections such as TREC CAsT, or quickly deploy interactive end-to-end systems through ParlAI.

Our framework is built on top of three major building blocks, shown in Figure 1. Here, we describe a multi-stage conversational passage retrieval system described in the earlier work of Lin et al. [9]: (a) the rewriter, which converts conversational queries into self-contained queries; (b) the retriever, which retrieves passage candidates from a collection of documents; and (c) the reranker, which rerankers the output from the retriever to produce better results. While other architectures are certainly possible, this is perhaps the most common design for conversational search systems today, and it is the one we have codified in our framework.

Rewriter. A conversational search system is expected to understand colloquial and contextually dependent utterances from users. In order to take advantage of existing *ad hoc* text retrieval techniques, one standard approach is to rewrite user utterances into clear and informative self-contained queries, based on the current utterance and those from previous turns in the conversation history. This is typically referred to as conversational query reformulation (CQR). Currently, we support two conversational query reformulation models under the CQR interface in Chatty Goose: Historical Query Expansion (HQE) and Neural Transfer Reformulation (NTR). The HQE model relies on the BM25 scoring function [16] to extract keywords for query expansion, whereas the NTR model leverages the pretrained text-to-text transfer transformer (T5) [14] to perform query rewrites. We implemented the HQE model using Pyserini and the NTR model using HuggingFace's Transformers library. Details about these two approaches are described in Lin et al. [9].

Retriever. The main purpose of a conversational search system is to lead users to relevant content that best satisfies their information needs, drawn from a collection of documents (the corpus). We provide retrieval functionality via Pyserini [8], which implements Python bindings for the Anserini IR toolkit [22]. Pyserini provides pre-built indexes for many standard test collections that can be directly downloaded, such as the test collection for TREC 2019 CAsT (abbreviated as CAsT19 hereafter). In our reference implementation, first-stage retrieval is performed using BM25 [16] through Pyserini's `SimpleSearcher` interface. While we provide a standard pipeline crafted for CAsT19, users can also leverage Pyserini to query their own collections.

Reranker. As in a standard multi-stage ranking architecture, first-stage retrieval results from the retriever above are reranked using pretrained neural models provided by PyGaggle,[1] a neural text ranking library that our group has developed. By default, we use the monoBERT (large, uncased) model provided by Nogueira and Cho [12], fine-tuned on the MS MARCO passage dataset [2]. This model has been shown to produce competitive results in large-scale passage ranking tasks [2, 6, 12].

3 IMPLEMENTATION AND USAGE

We have built Chatty Goose as a Python package `chatty-goose`, distributed via PyPI to allow for convenient installation via `pip`.[2] The main goal of our framework is to provide an easy way to create end-to-end conversational search systems, so we provide the `RetrievalPipeline` class as an abstraction over the conversational search pipeline. Each pipeline is defined by the three building blocks shown in Figure 1: one or more query rewriters, retriever,

[1] https://pypi.org/project/pygaggle
[2] https://pypi.org/project/chatty-goose

and reranker. Users can set up a full retrieval pipeline by providing all three components. When using multiple query rewriters, the pipeline module supports fusion using reciprocal rank fusion (RRF) [4], either after retrieval and reranking ("late fusion") or after retrieval but before reranking ("early fusion"). We refer readers to our earlier work [9] for a detailed comparison. We provide implementations for the HQE and NTR (T5) query reformulation methods as well as an abstract CQR class, where users can define the abstract method `rewrite` for conversational query reformulation. The other method is `reset_history` to reset the conversational history when beginning a new conversational session.

Chatty Goose can be installed using a single command via PyPI:

```
$ pip install chatty-goose==0.2.0
```

Here, we explicitly specify v0.2.0, which is presently our most recent release. Readers are advised, however, to use the latest version available. The following code snippet illustrates how one can quickly instantiate a `RetrievalPipeline` using components we provide:

```
# Creating the retrieval module for CAsT 2019
from pyserini.search import SimpleSearcher
searcher = SimpleSearcher.from_prebuilt_index("cast2019")
searcher.set_bm25(0.82, 0.68)

# Creating CQR modules
from chatty_goose.cqr import Hqe, Ntr
from chatty_goose.settings import HqeSettings, NtrSettings
hqe = Hqe(searcher, HqeSettings())
ntr = Ntr(NtrSettings())

# Creating the reranker module
from chatty_goose.util import build_bert_reranker
reranker = build_bert_reranker()

# Putting the modules together
from chatty_goose.pipeline import RetrievalPipeline
rp = RetrievalPipeline(searcher,
                       [hqe, ntr],
                       searcher_num_hits=1000,
                       early_fusion=True,
                       reranker=reranker)
```

With the above `RetrievalPipeline`, we can feed conversational queries to the system as follows:

```
turn1_results = rp.retrieve("What is throat cancer?")
turn2_results = rp.retrieve("Is it treatable?")

# End the current session and start a new one:
rp.reset_history()

# Examine the results:
for result in turn1_results:
    print(result.docid)
    print(result.raw)
```

Additionally, both CQR modules can be used in isolation to perform query reformulation, with an example using NTR (T5) shown below. It is also possible to do this directly using the HuggingFace Transformers library by simply loading our public model.[3]

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

```
turn1_text = ntr.rewrite("Tell me about the benefits of yoga.")
turn2_text = ntr.rewrite("Does it help in reducing stress?")
ntr.reset_history()
```

As shown above, we can use the CQR module (T5) to rewrite queries. The conversation history is cached in the CQR module, and can be purged by calling `reset_history()`.

4 EXPERIMENTS

Table 1 reports the effectiveness of three different Chatty Goose configurations on the evaluation set of the TREC 2019 Conversational Assistance Track (CAsT19) in terms of standard metrics. The first block of the table shows query rewriting using HQE and NTR (T5), as well as their combination using early fusion (which is more effective than late fusion). We follow the settings discussed in Section 3, using BM25 and BERT-large as our retriever and reranker, respectively.

	R@1000	MAP	NDCG@3
Chatty Goose runs			
HQE	.732	.306	.475
NTR (T5)	.739	.356	.546
Early fusion: HQE + NTR (T5)	.803	.374	.564
Original Results from Lin et al. [9]			
HQE	.730	.304	.481
NTR (T5)	.744	.359	.556
Early fusion: HQE + NTR (T5)	.804	.375	.565
Reference Comparisons			
Best CAsT '19 entry	-	.267	.436
QuReTec (RRF) [18]	-	.355	.476
Few-Shot Rewriter [23]	-	-	.492
Transformer++ [17]	-	.341	.529
MVR [7]	-	-	.565

Table 1: Results for full ranking with BM25 retrieval and BERT reranking on the CAsT19 evaluation set.

As we have explained, Chatty Goose implements techniques originally introduced in Lin et al. [9], shown in the second block of Table 1 for reference. Although there are minor differences in effectiveness, we believe that Chatty Goose successfully reproduces the figures reported in the earlier work. Due to different spaCy versions and implementation differences between TensorFlow (used in the original paper) and PyTorch (used here), it is not possible to reproduce the same results *exactly*. The third block of Table 1 presents a few other points of comparison. The best CAsT19 entry corresponds to our own work [21], which is the precursor to Lin et al. [9]. The remaining rows list the effectiveness of a few other systems that we are aware of on the same evaluation set. We can see that Chatty Goose provides a strong foundation for conversational search that others can build on. Table 2 presents an example of query reformulations using the HQE and the NTR (T5) modules on a session from CAsT19. We provide raw queries from earlier turns as context and show a comparison of the queries after rewriting, as well as the recall for each query.

Turn	Raw queries	HQE	NTR (T5)
1–5	*First 5 raw queries for context*: (1) What is Darwin's theory in a nutshell? (2) How was it developed? (3) How do sexual and asexual reproduction affect it? (4) How can fossils be used to understand it? (5) What is modern evidence for it?		
6	What is the impact on modern biology?	Darwin theory nutshell sexual asexual reproduction fossils evidence impact biology What is the impact on modern biology?	What is the impact of Darwin's theory on modern biology?
R@1000	0.186	0.576	0.830
7	Compare and contrast microevolution and macroevolution.	Darwin theory nutshell sexual asexual reproduction fossils evidence impact biology contrast microevolution macroevolution Compare and contrast microevolution and macroevolution.	Compare and contrast microevolution and macroevolution in Darwin's theory.
R@1000	0.962	0.981	0.887
8	What is the relationship to speciation?	Darwin theory nutshell sexual asexual reproduction fossils evidence impact biology contrast microevolution macroevolution relationship speciation What is the relationship to speciation?	What is Darwin's theory's relationship to speciation?
R@1000	0.500	0.868	0.842

Table 2: Reformulated queries for TREC CAsT19 session 56.

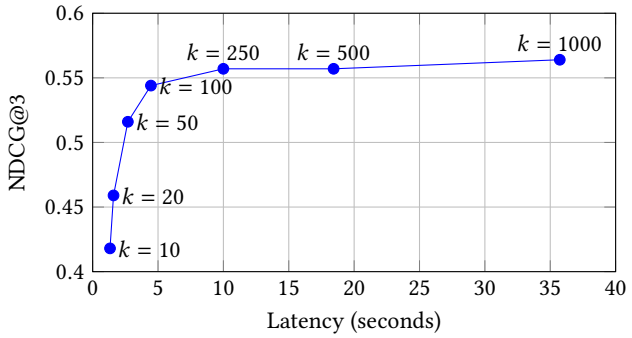

Figure 2: Tradeoff between latency and NDCG@3 in terms of k, the number of documents retrieved and reranked.

One important consideration for deploying production-ready systems is end-to-end latency. Figure 2 illustrates the tradeoff between efficiency and effectiveness, where we set the number of hits retrieved and reranked by our system, k, to different values and compare latency to NDCG@3. Measurements were run using Google Colab on a Tesla P100 GPU and a 2-core Intel Xeon CPU running at 2.3GHz. Latency was measured by taking the average query latency for the entire CAsT19 evaluation set.

5 TALKING TO CHATTY GOOSE

Chatty Goose provides an interactive conversational search agent that can be run out of the box. This agent integrates our components with ParlAI [11] to provide a command-line interface, which can be started with a simple command (with additional options to specify the index, CQR modules, etc.):

```
$ python -m chatty_goose.agents.chat
```

Another important goal of Chatty Goose is to simplify the process of deploying conversational agents to production environments. To demonstrate this capability, we provide an example server implementation and the necessary configuration for our agent to directly interface with Facebook Messenger, one of several chat service platforms supported by ParlAI. An example conversation using a CAsT19 session is shown in Figure 3. To provide a reasonable tradeoff between response latency and retrieval effectiveness, here we set the number of hits retrieved and reranked to a default value of 50, which gives an average latency of 2.7s per query, and an NDCG@3 of 0.516, based on Figure 2.

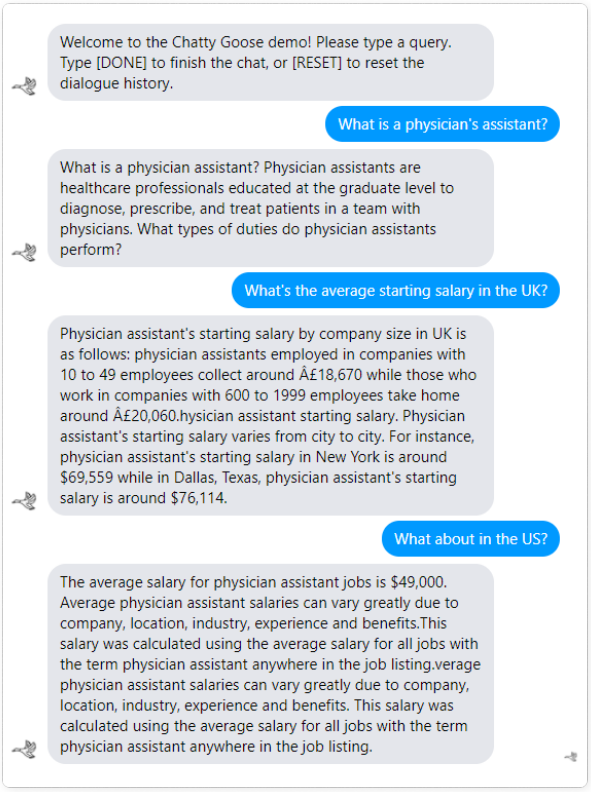

Figure 3: A screenshot of a Chatty Goose agent deployed as a Facebook Messenger chatbot using ParlAI.

6 CONCLUSIONS

As conversational search technology advances, it is important for systems to be flexible and accessible to the wider community. Chatty Goose is an open-source framework that allows researchers and practitioners to build, evaluate, and iterate on multi-stage conversational search pipelines in a reproducible manner. By providing infrastructure for researchers and practitioners to build on top of, we believe that Chatty Goose advances these goals.

ACKNOWLEDGEMENTS

This research was supported in part by the Canada First Research Excellence Fund, the Natural Sciences and Engineering Research Council (NSERC) of Canada. We would also like to thank Google for computational resources in the form of Google Cloud credits.

REFERENCES

[1] Nima Asadi and Jimmy Lin. 2013. Effectiveness/Efficiency Tradeoffs for Candidate Generation in Multi-Stage Retrieval Architectures. In *Proc. SIGIR*. 997–1000.

[2] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. MS MARCO: A human generated MAchine Reading COmprehension dataset. *arXiv:1611.09268* (2016).

[3] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question Answering in Context. In *Proc. EMNLP*. 2174–2184.

[4] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Buettcher. 2009. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. In *Proc. SIGIR*. 758–759.

[5] Jeffrey Dalton, Chenyan Xiong, and Jamie Callan. 2019. CAsT 2019: The Conversational Assistance Track Overview. In *Proc. TREC*.

[6] Laura Dietz and Nick Craswell. 2018. TREC Complex Answer Retrieval Overview. *Proc. TREC*.

[7] Vaibhav Kumar and Jamie Callan. 2020. Making Information Seeking Easier: An Improved Pipeline for Conversational Search. In *Proc. EMNLP: Findings*. 3971–3980.

[8] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proc. SIGIR*.

[9] Sheng-Chieh Lin, Jheng-Hong Yang, Rodrigo Nogueira, Ming-Feng Tsai, Chuan-Ju Wang, and Jimmy Lin. 2021. Multi-Stage Conversational Passage Retrieval: An Approach to Fusing Term Importance Estimation and Neural Query Rewriting. *ACM Trans. Inf. Syst.* (2021). In Press.

[10] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High Accuracy Retrieval with Multiple Nested Ranker. In *Proc. SIGIR*. 437–444.

[11] Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. ParlAI: A Dialog Research Software Platform. In *Proc. EMNLP: System Demonstrations*. 79–84.

[12] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv:1901.04085* (2019).

[13] Gustavo Penha, Alexandru Balan, and Claudia Hauff. 2019. Introducing MANtIS: A Novel Multi-Domain Information Seeking Dialogues Dataset. *arXiv:1912.04639* (2019).

[14] 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. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.

[15] Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. *Trans. ACL* 7 (March 2019), 249–266.

[16] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Proc. TREC*.

[17] Svitlana Vakulenko, Shayne Longpre, Zhucheng Tu, and Raviteja Anantha. 2021. Question Rewriting for Conversational Question Answering. In *Proc. WSDM*. 355–363.

[18] Nikos Voskarides, Dan Li, Pengjie Ren, Evangelos Kanoulas, and Maarten de Rijke. 2020. Query Resolution for Conversational Search with Limited Supervision. In *Proc. SIGIR*.

[19] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A Cascade Ranking Model for Efficient Ranked Retrieval. In *Proc. SIGIR*. 105–114.

[20] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proc. EMNLP: System Demonstrations*. 38–45.

[21] Jheng-Hong Yang, Sheng-Chieh Lin, Chuan-Ju Wang, Jimmy Lin, and Ming-Feng Tsai. 2019. Query and Answer Expansion from Conversation History. In *Proc. TREC*.

[22] Peilin Yang, Hui Fang, and Jimmy Lin. 2018. Anserini: Reproducible Ranking Baselines Using Lucene. *JDIQ* 10, 4 (2018), 16.

[23] Shi Yu, Jiahua Liu, Jingqin Yang, Chenyan Xiong, Paul Bennett, Jianfeng Gao, and Zhiyuan Liu. 2020. Few-Shot Generative Conversational Query Rewriting. In *Proc. SIGIR*. 1933–1936.

[24] Hamed Zamani and Nick Craswell. 2019. Macaw: An Extensible Conversational Information Seeking Platform. *arXiv:1912.08904* (2019).