Building More Reliable and Scalable AI Systems with Language Model Programming

Research Statement

Omar Khattab
okhattab@stanford.edu

Language models (LMs), used as building blocks within larger programs, are transforming the way we build AI systems. Despite overwhelming investment into "aligning" LMs as standalone generalist systems, user-facing LMs consistently fabricate statements and make fundamental reasoning errors, while imposing enormous costs. My work establishes **language model programming**, an alternative paradigm for building reliable and scalable AI systems. In it, we build multi-step programs that leverage *retrieval models* and LMs as modules, which we assign well-scoped but fuzzy computations like retrieval, decomposition, synthesis, and scoring. We can then *compile* these programs, i.e. automatically translate them into optimized strategies for prompting or finetuning LMs, to maximize system-level quality and minimize cost. In this way and toward it, my research has advanced the state of the art for document retrieval, question answering, fact checking, informative dialogue, and other knowledge-intensive Natural Language Processing (NLP) tasks.

I introduced *late interaction*, a popular paradigm for information retrieval (IR) that scales up LM *token-level attention* to entire document collections (**ColBERT**; **ColBERTv2**) and built state-of-the-art neural retrieval algorithms and systems that search millions of documents in milliseconds (**PLAID**; **DeepImpact**). Leveraging these, I designed reliable *retrieval-augmented* LM architectures (**Baleen**; **Hindsight**), which search massive collections to craft knowledgeable responses, and created new approaches for *supervising* retrievers and LMs (**ColBERT-QA**; **UDAPDR**) without human labels. I then proposed several new design patterns and programming abstractions (**Demonstrate–Search–Predict**) for building even more powerful *LM programs*, ones that resemble deep neural network architectures but rely on passing *text* (instead of tensors) between models (instead of individual neural layers). I then developed the first NLP compiler (**DSPy**) with automatic *supervision and optimization strategies* for LM programs. Instead of brittle "prompt engineering" or hand-crafted finetuning jobs, this compiler automates strategies for translating high-level LM programs into optimized chains of prompts (or LM weight updates) that maximize system-level objectives for any given LM.

My work spans several layers of the ML, NLP, and IR systems stacks: from efficient systems and algorithms to effective model architectures, supervision strategies, and programming abstractions. I prioritize building scalable systems that can unlock next-generation waves of ML applications. I regularly consolidate many core aspects of my research papers into two influential open-source systems: the **ColBERT** retrieval system (colbert.ai) and the **DSPy** programming framework (dspy.ai). Attracting hundreds of thousands of downloads and visitors per month, these platforms provide immediate feedback on my new research and a constant stream of cutting-edge ML problems from my users. My work forms the basis of research and applications at Google, Meta, Amazon, IBM, VMware, Baidu, Huawei, AliExpress, Vespa (formerly at Yahoo), Hebbia, Pointable, and You.com, to name a few in industry and is used by many academic teams at Stanford, Berkeley, CMU, MIT, Waterloo, UMD, Johns Hopkins, Cambridge, Glasgow, Amsterdam, and Tsinghua, among others.

1 Scaling up Expressive Neural Retrieval

LM programs require access to attributed and up-to-date information. I started my Ph.D. by developing the **ColBERT** [2] neural retrieval model, which introduced a new paradigm for Transformer-based retrieval called **late interaction** with previously unmatched quality and efficiency. If cost was not a concern, we might search millions of documents using an LM that *jointly* reads the query and each document together to measure relevance. This is known to be highly effective, as joint attention can model complex patterns (or *interactions*) between keywords in the query and document. However, this approach is extremely expensive, as the Transformer layers have to re-encode all documents for each new query. Because of that, most retrievers are *single-vector models*: they independently collapse each document (or query) into a single high-dimensional vector, and then measure relevance using just one dot product. This is highly scalable; it can search massive document collections in milliseconds. Unfortunately, it also creates a substantial bottleneck that limits these models' ability to learn complex interactions. My late interaction paradigm seeks to resolve this tension: it proposes effective and massively scalable methods to permit attention-like interactions between queries and documents. ColBERT independently encodes each document (or query) as a token-level *bag of vectors* and then estimates relevance with

"MaxSim" operators, a sum of maximum-similarity scores. These operators act as a fast form of attention between two bags of vectors, and manage to preserve the *retrieval precision* of BERT rankers based on cross-attention, while being *orders-of-magnitude* cheaper as late interaction permits encoding all documents offline.

ColBERT established a new Pareto frontier in quality and cost: it can scale *sub-linearly*, searching millions of documents in milliseconds, by being *pruning-friendly*. ColBERT leverages the mathematical properties of MaxSim to quickly skip more than 99.9% of the (low-scoring) documents when retrieving the top-k results. Essentially, it can decompose late interaction into dozens of small nearest-neighbor searches at the token level, fetching and scoring only documents in which at least one token is close to (at least one token in) the query. Otherwise, we can prove the total score will be too small, and we can skip it. These massive cost reductions enabled ColBERT to raise the *retrieval recall* dramatically, e.g. to 97% by ColBERT instead of 81% by the BM25 system re-ranked by BERT models. A few months after I conducted this work, Meta released their popular single-vector DPR [25] retriever. In ColBERT-QA [6], I compared the two: I found that a *zero-shot* ColBERT retriever can outperform *specialized* DPR retrievers trained on *tens of thousands of in-domain queries* for the test domain. When both are trained similarly, ColBERT can outperform DPR by up to 19 points in answer recall.

I designed **ColBERTv2** [14], which further improves the quality of ColBERT embeddings and introduces a powerful *residual compression* technique. Based on a new hypothesis of why ColBERT embeddings work well, this technique cuts ColBERT's index size by an order of magnitude, consuming as little as *20 bytes per vector*. On 23 of 29 test tasks, ColBERTv2 achieved the highest quality of any standalone retriever, outperforming the next best retriever by up to 8%, while using its compressed representations. I later designed **PLAID** [13], a retrieval engine that leverage these residual representations to speed up late interaction dramatically. PLAID can search 140,000,000 passages in around 100 milliseconds. In July 2023, I was invited to give a keynote at the SIGIR ReNeuIR workshop. In my keynote, I highlighted how these IR–NLP–Systems efforts helped overcome the "hardware lottery" [23], in which superior but unconventional ML architectures are overlooked due to efficiency concerns. Throughout 2023, we have hosted a public ColBERTv2 index of Wikipedia, in which PLAID has served millions of queries toward papers by multiple teams on retrieval-augmented LMs. Further out on the efficiency side, late interaction has created cross-disciplinary work even beyond CS, including ongoing collaborations with Stanford electrical engineers and neuroscientists on hardware chip design for retrieval.

With 1500 citations in a short time for my ColBERT line of work, which includes also **ColBERTer** [1], **UDAPDR** [12], and other systems, it has left considerable impact on IR and NLP, in academia and industry. Dozens of teams have built new models and products atop ColBERT models [18; 21; 29; 30; 22] or extending them [44; 37; 40; 35; 27]. Tens of studies have shown that late interaction surpasses the quality of single-vector models in out-of-domain retrieval settings [39; 33; 42], challenging domains like math and literature [43; 38; 31], cross-language retrieval [28; 34; 19], and even multi-modal retrieval and visual question answering [41; 32]. Papers *building on ColBERT* by several teams have become highly influential: work analyzing [20] and extending [26] ColBERT has received best paper awards at top venues, and work applying late interaction in Computer Vision [41] has received over 450 citations. A recent paper [16] from TU Wien found that ColBERT was particularly data efficient: with 1000 training queries, ColBERT not only outperforms expensive cross-encoders but also matches single-vector DPR models trained with 50× more queries! Another recent paper found that ColBERT trained for Japanese [19] can match or outperform single-vector models trained on about 100× more data.

2 Building Reliable LM Systems via Retrieval

Language models struggle with recalling long-tail facts and often hallucinate statements or make up citations. Their knowledge is also hard to update or specialize. To deal with this, I built several early *retrieval-based* LM systems [4], which decouple the LM's capacity for *processing text* from how it *stores knowledge*. This boosts reliability and trust: when these systems produce responses, we can read the sources they cite and judge their relevance and credibility [3] and, since they store knowledge as text, their LMs can be small and cheap [5] and their knowledge can be efficiently updated or expanded. A key challenge is how to train the system's intermediate stages, for which we typically lack human labels. I developed **ColBERT-QA** [6], a question answering (QA) system that introduces *relevance-guided supervision* (RGS), an iterative strategy for training retrievers without retrieval labels. RGS takes a noisy heuristic from a downstream task like QA (e.g., "does a retrieved passage contain the right answer?") and proceeds in several rounds, training the retriever to assign higher scores to the *most retrievable* documents that pass the heuristic. This simple approach, later generalized in several projects, proved more scalable and more expressive than existing alternatives: I showed that applying RGS to ColBERT, yielding ColBERT-QA, increases answer recall significantly, without any retrieval labels, and produces a QA system whose answer accuracy exceeds other systems by up to 12 points. This materialized against much larger LMs: on the NaturalQuestions benchmark, ColBERT-QA achieves 48% EM, whereas fine-tuned T5-11B [36] (24x larger) and few-shot GPT-3 [17] (400x larger) score 35% and 30%, respectively.

Focusing next on more complex reasoning tasks, I developed **Baleen** [5]. While a two-step *retrieve-then-generate* pipeline (nowadays referred to as "RAG") may answer simple questions, it fails when there are complex dependencies between facts. To deal with complex questions, the literature suggests applying multiple "hops" of search to gather information iteratively. However, doing so naively may drown the search system in an exponential space of potential search paths. I introduced *condensed retrieval*, a paradigm for conducting multiple retrieval hops in which the system reads tens of retrieved documents and *summarizes them in each hop* to inform its search trajectory. This new architecture achieved high scalability, by nature of maintaining one path via iterative summarization, and achieved exceedingly high recall, by reading many documents over several hops. However, supervising Baleen was challenging, as we do not know the order in which information dependencies should be resolved. I tackled this with a novel *latent hop ordering* (LHO) algorithm, which generalizes the ideas behind ColBERT-QA's RGS to multiple retrieval hops. Baleen advanced quality on HoVer [24], a challenging claim verification benchmark, from the official baseline's 15% score directly to 57%, approaching human-level quality. Overall, I have demonstrated through several state-of-the-art LM systems, including ColBERT-QA, Baleen, and later **Hindsight** [10], that leveraging language models as composable modules, not end-to-end systems, and creating new supervision strategies to train them in multi-stage architectures are both essential for building more reliable and efficient user-facing systems. My work in this space has led to significant applications at VMware, IBM, Hebbia, Virtusa, and others, some of which are among the earliest retrieval-based LM production systems.

3 Language Model Programming

Recent improvements for LMs have fueled an exploding space of "prompting" techniques, in which LMs can be *programmed* with little data using natural language instructions. To build multi-stage architectures in this regime, I created the **Demonstrate–Search–Predict** (DSP; [7]) abstractions, expressing a large space of new AI systems as text-exchange strategies between a *frozen* LM and a retriever. I introduced general primitives to annotate few-shot examples for any such strategy (Demonstrate), gather relevant knowledge from multiple sources (Search), and generate grounded predictions (Predict). I expressed new strategies with these primitives as DSP programs for a few tasks, showing they outperform GPT-3.5 and LM pipelines and agents by up to 120% and 40%, respectively. Interest in multi-stage LM pipelines has since grown dramatically. Existing work implements these with hard-coded 'prompt templates', i.e. long hand-crafted strings. This approach can be brittle and unscalable: a prompt might not generalize to different pipelines, LMs, or even inputs.

I created the **DSPy programming model** [8], which introduces declarative modules, akin to neural network layers, to abstract text transformations that are otherwise conducted by manual prompting or finetuning of LMs. Each DSPy module is *parameterized* so it can learn to conduct its behavior, to maximize a system-level objective. To leverage this, I developed the **DSPy compiler**, which takes any high-level DSPy program and produces an optimized chain of prompts (or LM finetuned weights). Given a metric and some training inputs, the DSPy compiler repeatedly simulates the program to bootstrap example traces of each module, using them for self-improvement, i.e. to construct effective few-shot prompts, create high-quality instructions, or finetune small LMs for the pipeline steps. Without hand-crafted prompts and within minutes of compiling, a few lines of DSPy allow LMs like `GPT-3.5` and `Llama2-13b` to self-bootstrap pipelines that outperform standard few-shot prompting (generally by over 25% and 65%, respectively) and pipelines with expert-created demonstrations (by up to 5–46% and 16–40%, respectively). On top of that, DSPy programs compiled to open and small LMs like `Llama2-13b` and `T5-770M` are competitive with approaches that rely on expert-written prompts for large proprietary LMs.

Even though DSPy is only a year old, it has a large and fast-growing user base, including in industry at Amazon, VMware, Microsoft, Replit, Google, Databricks, Pointable, JetBlue, Walmart, and You.com as well as in research groups at Stanford University (several teams), UC Berkeley, CMU, MIT, University of Waterloo, UMD, and many others. At the time of writing, the DSPy GitHub repository has nearly 14,000 stars and 1,100 forks, is visited by tens of thousands of unique users per month, and has nearly 180 open-source contributors. DSPy has been used by multiple teams to build best-in-class LM systems, including for biomedical information extraction, document retrieval, and drafting long articles with citations. This work has resonated deeply in industry and beyond: I have been invited to present this work at Apple, Google, Google X, Meta, Databricks, Oracle Labs, UC Berkeley, and SIGIR REML, among a dozen other venues. I have also been able to receive approximately $300,000 in grant funding for DSPy work from Oracle Labs, IBM Research AI, and HAI–Azure Cloud Credit Program.

4 Future Research

The paradigm of language model programming is just starting to transform AI systems. I often compare LM programs to the historical development of deep neural networks (DNNs). Both LM programs and DNNs compose modular "layers"—i.e., calls to LM modules and DNN operations—into larger architectures, ones that can be optimized in different ways toward objectives. This analogy spawns many interesting research questions and

applications. I plan to study these questions while continuing to build scalable NLP systems that spark next-generation waves of applications across NLP, IR, and ML Systems. Naturally, much of my work will leverage the active community around ColBERT and DSPy, but I plan to broaden my explorations into the following:

Universal LM Programs. While the DNN literature explored numerous task-specific architectures up to the 2010s, the ML community eventually converged on a few universal architectures (e.g., Transformers) and task-independent pretraining recipes (e.g., with language modeling objectives). How would this look for LM programs? I expect such a "universal" LM program will contain a large number of recursive structures, subsuming several redundant components for retrieval, grounded generation, classical planning, routing between experts, and the creation and caching of "skills" or demonstrations. This direction shares commonalities with *language agents* and with finetuning LMs to use tools. Unlike them, however, univeral LM programs will place emphasis on finding the right multi-stage architecture, with appropriate inductive biases, and will be optimized automatically (through bootstrapping optimizers, like the DSPy compiler) directly toward multiple high-level objectives.

LM Program Optimizers. We can model the optimization process of a DSPy program as a reinforcement learning (RL) or a graph search problem, where an agent can act to tune select a module, to tune its instructions, bootstrap new examples for it, and/or finetune its LM weights. How can we train such RL agents to perform general LM program optimization? Toward this vision, I have mentored two projects with forthcoming releases on optimizing the instructions of prompts in DSPy and on integrating *DSPy assertions*, a general construct that facilitates automatic self-reflection and backtracking logic to enforce constraints on programs.

Richer Retrieval Models. Combining insights from DSPy and Baleen into ColBERT, future retrieval models will themselves be LM programs, thereby serving as instructable foundation models for retrieval. They will invoke an LM to generate candidate search queries, read the retrieve documents, and plan or refine its search trajectory accordingly. How can we make this efficient and scalable? How do we supervise this kind of approach so it reaches its potential? Toward this long-term vision, I have mentored two advanced-stage projects on building rerankers in DSPy and developing the next ColBERTv2.5 model.

Better Benchmarking Paradigms. NLP brenchmarks have not kept up with the progress in the field, in part due to poor incentives. Indeed, it appears easier now to build a system for a new task than to evaluate one. I have co-led five benchmarking efforts (including LoTTE [14], HELM's IR evals, and [15]) during my Ph.D., and gleaned several key lessons from them. Leveraging these, I plan to invest substantial research into benchmarks that (1) derive test queries from natural user interactions, (2) disentangle *head* topics vs. the *long tail*, (3) expose "zero-shot" challenges, in which no training or validation data is supplied, but nonetheless (4) feature a separate distribution for validation as in [14], to simultaneously facilitate active development and disincentivize overfitting. In such benchmarks, metrics must account for balancing the cost and quality dimensions as in [15]. Moving past classical text-generation metrics in NLP, automatic metrics should themselves consistent of LM programs for evaluation. To be able to trust these metrics, benchmarks must establish processes by which every evaluation effort (e.g., a paper reporting new scores) will report confidence intervals and contribute new data back into the metrics by collecting a small set of human comparisons, as in [11].

References to My Work

[1] S. Hofstatter, O. Khattab, S. Althammer, M. Sertkan, and A. Hanbury. Introducing neural bag of whole-words with ColBERTer: Contextualized late interactions using enhanced reduction. In *CIKM*, 2022.
[2] O. Khattab and M. Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *SIGIR*, 2020.
[3] O. Khattab, C. Potts, and M. Zaharia. A moderate proposal for radically better AI-powered Web search. Stanford HAI Blog, 2021.
[4] O. Khattab, C. Potts, and M. Zaharia. Building scalable, explainable, and adaptive NLP models with retrieval. Stanford AI Lab Blog, 2021.
[5] O. Khattab, C. Potts, and M. Zaharia. Baleen: Robust Multi-Hop Reasoning at Scale via Condensed Retrieval. In *NeurIPS*, 2021.
[6] O. Khattab, C. Potts, and M. Zaharia. Relevance-guided supervision for openqa with ColBERT. *Transactions of the Association for Computational Linguistics*, 9:929–944, 2021.
[7] O. Khattab, K. Santhanam, X. L. Li, D. Hall, P. Liang, C. Potts, and M. Zaharia. Demonstrate-Search-Predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv*, 2022.
[8] O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T Joshi, H. Moazam, H. Miller, M. Zaharia, and C. Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. *arXiv*, 2023.
[9] A. Mallia, O. Khattab, T. Suel, and N. Tonellotto. Learning passage impacts for inverted indexes. In *SIGIR*, pages 1723–1727, 2021.
[10] A. Paranjape, O. Khattab, C. Potts, M. Zaharia, and C. D. Manning. Hindsight: Posterior-guided Training of Retrievers for Improved Open-ended Generation. In *ICLR*, 2022.
[11] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia. ARES: An automated evaluation framework for retrieval-augmented generation systems. *arXiv*, 2023.
[12] J. Saad-Falcon, O. Khattab, K. Santhanam, R. Florian, M. Franz, S. Roukos, A. Sil, M. A. Sultan, and C. Potts. UDAPDR: Unsupervised domain adaptation via llm prompting and distillation of rerankers. *EMNLP*, 2023.
[13] K. Santhanam*, O. Khattab*, C. Potts, and M. Zaharia. PLAID: An Efficient Engine for Late Interaction Retrieval. *CIKM*, 2022.
[14] K. Santhanam*, O. Khattab*, J. Saad-Falcon, C. Potts, and M. Zaharia. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. *NAACL*, 2022.
[15] K. Santhanam, J. Saad-Falcon, M. Franz, O. Khattab, A. Sil, R. Florian, M. A. Sultan, S. Roukos, M. Zaharia, and C. Potts. Moving beyond downstream task accuracy for information retrieval benchmarking. *ACL Findings*, 2023.

Other References

[16] Sophia Althammer, Guido Zuccon, Sebastian Hofstätter, Suzan Verberne, and Allan Hanbury. Annotating data for fine-tuning a neural ranker? current active learning strategies are not better than random selection. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 139–149, 2023.

[17] Tom Brown et al. Language models are few-shot learners. *NeurIPS*, 2020.

[18] Jaekeol Choi, Euna Jung, Jangwon Suh, and Wonjong Rhee. Improving bi-encoder document ranking models with two rankers and multi-teacher distillation. *arXiv:2103.06523*, 2021.

[19] Benjamin Clavié. Jacolbert and hard negatives, towards better japanese-first embeddings for retrieval: Early technical report. *arXiv preprint arXiv:2312.16144*, 2023.

[20] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. A white box analysis of colbert. In *43rd European Conference on Information Retrieval (ECIR)*, 2021.

[21] Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. Improving Efficient Neural Ranking Models with Cross-Architecture Knowledge Distillation. *arXiv preprint arXiv:2010.02666*, 2020. URL https://arxiv.org/abs/2010.02666.

[22] Sebastian Hofstätter, Sheng-Chieh Lin, Jheng-Hong Yang, Jimmy Lin, and Allan Hanbury. Efficiently Teaching an Effective Dense Retriever with Balanced Topic Aware Sampling. *arXiv preprint arXiv:2104.06967*, 2021. URL https://arxiv.org/abs/2104.06967.

[23] Sara Hooker. The hardware lottery. *Communications of the ACM*, 64(12):58–65, 2021.

[24] Yichen Jiang et al. HoVer: A dataset for many-hop fact extraction and claim verification. *Findings of EMNLP*, 2020.

[25] Vladimir Karpukhin et al. Dense passage retrieval for open-domain question answering. In *EMNLP*, Online, 2020.

[26] Weize Kong, Jeffrey M Dudek, Cheng Li, Mingyang Zhang, and Michael Bendersky. Sparseembed: Learning sparse lexical representations with contextual embeddings for retrieval. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2399–2403, 2023.

[27] Jinhyuk Lee, Zhuyun Dai, Sai Meher Karthik Duddu, Tao Lei, Iftekhar Naim, Ming-Wei Chang, and Vincent Y Zhao. Rethinking the role of token retrieval in multi-vector retrieval. *arXiv preprint arXiv:2304.01982*, 2023.

[28] Yulong Li, Martin Franz, Md Arafat Sultan, Bhavani Iyer, Young-Suk Lee, and Avirup Sil. Learning cross-lingual ir from an english retriever. *arXiv preprint arXiv:2112.08185*, 2021.

[29] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Distilling Dense Representations for Ranking using Tightly-Coupled Teachers. *arXiv preprint arXiv:2010.11386*, 2020. URL https://arxiv.org/abs/2010.11386.

[30] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. Contextualized query embeddings for conversational search. *arXiv:2104.08707*, 2021.

[31] Weizhe Lin, Rexhina Blloshmi, Bill Byrne, Adrià de Gispert, and Gonzalo Iglesias. Li-rage: Late interaction retrieval augmented generation with explicit signals for open-domain table question answering. 2023.

[32] Weizhe Lin, Jinghong Chen, Jingbiao Mei, Alexandru Coca, and Bill Byrne. Fine-grained late-interaction multi-modal retrieval for retrieval augmented visual question answering. *arXiv preprint arXiv:2309.17133*, 2023.

[33] Simon Lupart, Thibault Formal, and Stéphane Clinchant. Ms-shift: An analysis of ms marco distribution shifts on neural retrieval. In *European Conference on Information Retrieval*, pages 636–652. Springer, 2023.

[34] Suraj Nair, Eugene Yang, Dawn Lawrie, Kevin Duh, Paul McNamee, Kenton Murray, James Mayfield, and Douglas W Oard. Transfer learning approaches for building cross-language dense retrieval models. In *European Conference on Information Retrieval*, pages 382–396. Springer, 2022.

[35] Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y Zhao. Multi-vector retrieval as sparse alignment. *arXiv preprint arXiv:2211.01267*, 2022.

[36] Adam Roberts et al. How much knowledge can you pack into the parameters of a language model? In *EMNLP*, 2020.

[37] Weng Lam Tam, Xiao Liu, Kaixuan Ji, Lilong Xue, Xingjian Zhang, Yuxiao Dong, Jiahua Liu, Maodi Hu, and Jie Tang. Parameter-efficient prompt tuning makes generalized and calibrated neural text retrievers. *arXiv preprint arXiv:2207.07087*, 2022.

[38] Katherine Thai, Yapei Chang, Kalpesh Krishna, and Mohit Iyyer. Relic: Retrieving evidence for literary claims. *arXiv preprint arXiv:2203.10053*, 2022.

[39] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. *arXiv preprint arXiv:2104.08663*, 2021. URL https://arxiv.org/abs/2104.08663.

[40] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. Colbert-prf: Semantic pseudo-relevance feedback for dense passage and document retrieval. *ACM Transactions on the Web*, 17(1):1–39, 2023.

[41] Lewei Yao, Runhui Huang, Lu Hou, Guansong Lu, Minzhe Niu, Hang Xu, Xiaodan Liang, Zhenguo Li, Xin Jiang, and Chunjing Xu. Filip: Fine-grained interactive language-image pre-training. *arXiv preprint arXiv:2111.07783*, 2021.

[42] Jingtao Zhan, Xiaohui Xie, Jiaxin Mao, Yiqun Liu, Jiafeng Guo, Min Zhang, and Shaoping Ma. Evaluating interpolation and extrapolation performance of neural retrieval models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 2486–2496, 2022.

[43] Wei Zhong, Jheng-Hong Yang, Yuqing Xie, and Jimmy Lin. Evaluating token-level and passage-level dense retrieval models for math information retrieval. *arXiv preprint arXiv:2203.11163*, 2022.

[44] Giulio Zhou and Jacob Devlin. Multi-vector attention models for deep re-ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5452–5456, 2021.