PrimeQA: The Prime Repository for State-of-the-Art Multilingual Question Answering Research and Development

Avirup Sil[*], Jaydeep Sen, Bhavani Iyer, Martin Franz, Kshitij Fadnis, Mihaela Bornea, Sara Rosenthal, Scott McCarley, Rong Zhang, Vishwajeet Kumar, Yulong Li, Md Arafat Sultan, Riyaz Bhat, Juergen Bross, Radu Florian, Salim Roukos
IBM Research AI

Abstract

The field of Question Answering (QA) has made remarkable progress in recent years, thanks to the advent of large pre-trained language models, newer realistic benchmark datasets with leaderboards, and novel algorithms for key components such as retrievers and readers. In this paper, we introduce PrimeQA: a one-stop and open-source QA repository with an aim to democratize QA research and facilitate easy replication of state-of-the-art (SOTA) QA methods. PrimeQA supports core QA functionalities like retrieval and reading comprehension as well as auxiliary capabilities such as question generation. It has been designed as an end-to-end toolkit for various use cases: building front-end applications, replicating SOTA methods on public benchmarks, and expanding pre-existing methods. PrimeQA is available at: https://github.com/primeqa.

1 Introduction

Question Answering (QA) is a major area of investigation in Natural Language Processing (NLP), consisting primarily of two subtasks: information retrieval (IR) (Manning, 2008; Schütze et al., 2008) and machine reading comprehension (MRC) (Rajpurkar et al., 2016, 2018; Kwiatkowski et al., 2019a; Chakravarti et al., 2020). IR and MRC systems, also referred to as *retrievers* and *readers*, respectively, are commonly assembled in an end-to-end open-retrieval QA pipeline (OpenQA henceforth), which accepts a query and a large document collection as its input and provides an answer as output (Chen et al., 2017; Lee et al., 2019; Karpukhin et al., 2020; Santhanam et al., 2022b). The retriever first identifies documents or passages (i.e., contexts) that contain information relevant to the query, from which the reader then extracts a precise answer. Alternatively, the reader can also be generative and leverage large language models (LLMs) (Ouyang et al., 2022; Chung et al., 2022).

Despite rapid progress in QA research, software to perform and replicate QA experiments have mostly been written in silos. At the time of this writing, there is no central repository that facilitates the training, analysis and augmentation of state-of-the-art (SOTA) models for different QA tasks at scale. In view of the above, and with an aim to democratize QA research by providing easy replicability, here we present PrimeQA: an open-source repository[1] designed as an end-to-end toolkit. It offers all the necessary tools to easily and quickly create a custom QA application. We provide a main repository that contains easy-to-use scripts for retrieval, machine reading comprehension, and question generation with the ability to perform training, inference, and performance evaluation. Additionally, several sibling repositories offer features for easily connecting various retrievers and readers, as well as for creating a front-end user interface (UI). PrimeQA has been designed as a platform for QA development and research, and encourages collaboration from all members of the QA community—from beginners to experts. PrimeQA has a growing developer base with contributions from major academic institutions.

The following is a summary of our contributions:

- We present PrimeQA, a first-of-its-kind repository for comprehensive QA research. It is free to use, well-documented, easy to contribute to, and license-friendly (Apache 2.0) for both academic and commercial usage.
- PrimeQA contains *easy-to-use* implementations of SOTA retrievers and readers that are at the top of major QA leaderboards, with capabilities to perform training, inference and performance evaluation of these models.
- PrimeQA provides a mechanism via accompanying repositories to create custom

[*]Corresponding author: avi@us.ibm.com

[1]https://github.com/primeqa

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics Volume 3: System Demonstrations, pages 51–62
July 10-12, 2023 ©2023 Association for Computational Linguistics

OpenQA applications for industrial deployment, including a front-end UI.
- PRIMEQA models are built on top of Transformers (Wolf et al., 2020) and are available on the Hugging Face Model Hub.[2]
- PRIMEQA has readers that can leverage SOTA LLMs such as InstructGPT (Ouyang et al., 2022) via external APIs.

2 Related Work

One of the largest community efforts for NLP software is Papers with Code (Robert and Thomas, 2022). Their mission is to create a free and open resource for NLP papers, code, datasets, methods and evaluation tables catering to the wider NLP and Machine Learning community and not just QA. Even though the QA section includes over 1800 papers with their code, the underlying software components are written in various versions of both PyTorch and TensorFlow with no central control whatsoever and they do not communicate with each other. These disjoint QA resources hinder replicability and effective collaboration, and ultimately lead to quick "sunsetting" of new capabilities.

Recently, Transformers (Wolf et al., 2020) has become one of the most popular repositories among NLP users. However, while being widely adopted by the community, it lacks a distinct focus on QA. Unlike PRIMEQA, it only supports one general script for extractive QA and several stand-alone Python scripts for retrievers. Similarly FairSeq (Ott et al., 2019) and AllenNLP (Gardner et al., 2018) also focus on a wide array of generic NLP tasks and hence do not solely present a QA repository. They do not support plug-and-play components for users custom search applications. Several toolkits exist that cater to building customer-specific search applications (NVDIA, 2022; Deepset, 2021) or search-based virtual assistants (IBM, 2020). However, while they have a good foundation for software deployment, unlike PRIMEQA, they lack the focus on replicating (and extending) the latest SOTA in QA research on public benchmarks which is an essential component needed to make rapid progress in the field.

3 PRIMEQA

PRIMEQA is a comprehensive open-source resource for cutting-edge QA research and development, governed by the following design principles:

Core Models	Extensions
Retriever	
BM25 (Robertson and Zaragoza, 2009)	Dr.DECR * (Li et al., 2022)
DPR (Karpukhin et al., 2020)	
ColBERT (Santhanam et al., 2022b)	
Reader	
General MRC* (Alberti et al., 2019b)	ReasonBERT (2021)
FiD (Izacard and Grave, 2020)	OmniTab (Jiang et al., 2022a)
Boolean* (McCarley et al., 2023)	MITQA* (Kumar et al., 2021)
Lists	
Tapas (Herzig et al., 2020a)	
Tapex (Liu et al., 2021)	
Question Generation	
Table QG (Chemmengath et al., 2021)	
Passage QG	
Table+Passage QG	

Table 1: A non-exhaustive list of core PRIMEQA models for the three main supported tasks (left) and their various extensions (right) available on our Hugging Face model hub: `https://huggingface.co/PrimeQA`. * SOTA leaderboard systems.

- **Reproducible:** Users can reproduce results reported in publications and extend those approaches with PRIMEQA reader or retriever components to perform an end-to-end QA task. The PRIMEQA components are listed in Table 1.
- **Customizable:** We allow users to customize and extend SOTA models for their own applications. This often entails fine-tuning on users custom data.
- **Reusable:** We aim to make it straightforward for developers to quickly deploy pre-trained off-the-shelf PRIMEQA models for their QA applications, requiring minimal code change.
- **Accessible:** We provide easy integration with Hugging Face Datasets and the Model Hub, allowing users to quickly plug in a range of datasets and models as shown in Table 1.

PRIMEQA in its entirety is a collection of four different repositories: a primary *research and replicability*[3] repository and three accompanying repositories[4,5,6] for industrial deployment. Figure 1 shows a diagram of the PrimeQA repository. It provides several entry points, supporting the needs of different users, as shown at the top of the figure. The repository is centered around three core components: a **retriever**, a **reader**, and a **question generator** for data augmentation. These components can be used as individual modules or assembled

[2] https://huggingface.co/PrimeQA

[3] https://github.com/primeqa/primeqa

[4] https://github.com/primeqa/create-primeqa-app

[5] https://github.com/primeqa/primeqa-orchestrator

[6] https://github.com/primeqa/primeqa-ui

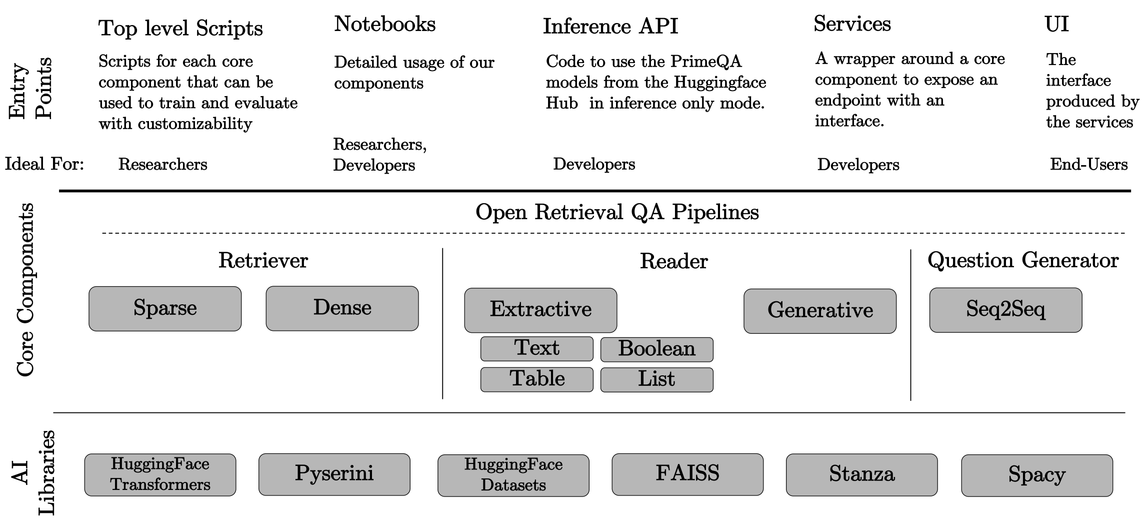

Figure 1: The PRIMEQA Repository: the core components and features.

into an end-to-end QA pipeline. All components are implemented on top of existing AI libraries.

3.1 The Core Components

Each of the three core PRIMEQA components supports different flavors of its corresponding task, as we detail in this section.

3.1.1 Retriever: `run_ir.py`

Retrievers predict documents (or passages) from a document collection that are relevant to an input question. PRIMEQA has both sparse and SOTA dense retrievers along with their extensions, as shown in Table 1. We provide a single Python script `run_ir.py` that can be passed arguments to switch between different retriever algorithms.

Sparse: BM25 (Robertson and Zaragoza, 2009) is one of the most popular sparse retrieval methods, thanks to its simplicity, efficiency and robustness. Our Python-based implementation of BM25 is powered by the open-source library PySerini.

Dense: Modern neural retrievers have utilized dense question and passage representations to achieve SOTA performance on various benchmarks, while needing GPUs for efficiency. We currently support ColBERT (Santhanam et al., 2022b) and DPR (Karpukhin et al., 2020): both fine-tune pre-trained language models to train question and passage encoders (Devlin et al., 2019; Conneau et al., 2020). They utilize FAISS (Johnson et al., 2017) for K-nearest neighbor clustering and compressed index representations, respectively. They support multilingual retrieval with the question and the documents being in the same (Lee et al., 2019; Longpre et al., 2021) or different languages (cross-lingual) (Asai et al., 2021).

3.1.2 Reader: `run_mrc.py`

Given a question and a retrieved passage—also called the *context*—a reader predicts an answer that is either extracted directly from the context or is generated based on it. PRIMEQA supports training and inference of both extractive and generative readers through a single Python script: `run_mrc.py`. It works out-of-the-box with different QA models extended from the Transformers library (Wolf et al., 2020).

Extractive: PRIMEQA's general extractive reader is a pointer network that predicts the start and end of the answer span from the input context (Devlin et al., 2019; Alberti et al., 2019b). It can be initialized with most large pre-trained language models (Devlin et al., 2019; Liu et al., 2019; Conneau et al., 2020). In addition, our reader is extremely versatile as it can provide responses to questions with list answers (Khashabi et al., 2021), *yes/no* responses to Boolean questions (Clark et al., 2019, 2020a; Kwiatkowski et al., 2019b), answer spans found in tables (Herzig et al., 2020b) and in multimodal (text+image) documents (Mathew et al., 2021). Examples of several extractive readers along with their extensions are provided in Table 1.

Generative: PRIMEQA provides generative readers based on the popular Fusion-in-Decoder (FiD) (Izacard and Grave, 2020) algorithm. Currently, it supports easy initialization with large pre-trained sequence-to-sequence models (Lewis et al., 2019; Raffel et al., 2022). With FiD, the question and the retrieved passages are used to generate relatively long and complex multi-sentence answers providing support for long form question answering tasks, *e.g.*, ELI5 (Petroni et al., 2021; Fan et al., 2019).

3.1.3 Question Generation: `run_qg.py`

Data augmentation through synthetic question generation (QG) helps in generalization of QA models (Alberti et al., 2019a; Sultan et al., 2020), especially when labeled data is not available for the target domain. It can be applied in a variety of settings, including domain adaptation (Shakeri et al., 2021; Gangi Reddy et al., 2021, 2022), domain generalization (Sultan et al., 2022) and few-shot learning (Yue et al., 2022). PRIMEQA's QG component (Chemmengath et al., 2021) is based on SOTA sequence-to-sequence generation architectures (Raffel et al., 2022), and supports both unstructured and structured input text through a single Python script `run_qg.py`.

Unstructured Input: Our first variant of QG is a multilingual text-to-text model capable of generating questions in the language of the input passage. It fine-tunes a pre-trained T5 language model (Raffel et al., 2022) on publicly available multilingual QA data (Clark et al., 2020b).

Structured Input: Our second variant learns QG over tables by fine-tuning T5 (Raffel et al., 2022) to generate natural language queries using the Table QA dataset (Zhong et al., 2017a). Given a table, PRIMEQA uses a controllable SQL sampler to obtain SQL queries and then applies the trained table QG model to generate natural language questions.

Semi-structured Input: PRIMEQA also supports QG over tables and text by fine-tuning T5 (Raffel et al., 2022) to generate natural language queries from table+text context. The training data is obtained using the publicly available HybridQA dataset (Chen et al., 2020).

3.2 Entry Points

We cater to different user groups in the QA community by providing different entry points to PRIMEQA, as shown in Figure 1.

- **Top-level Scripts:** Researchers can use the top level scripts, `run_{ir/mrc/qg}.py`, to reproduce published results and train, fine-tune and evaluate associated models on their own custom data.
- **Jupyter Notebooks:** These demonstrate how to use built-in classes to run the different PRIMEQA components and perform the corresponding tasks. They are useful for developers and researchers who want to reuse and extend PRIMEQA functionalities.
- **Inference APIs:** The Inference APIs are primarily meant for developers, allowing them to use PRIMEQA components on their own data with only a few lines of code. These APIs can be initialized with the pre-trained PRIMEQA models provided in the HuggingFace hub, or with a custom model that has been trained for a specific use case.
- **Service Layer:** The service layer helps developers set up an end-to-end QA system quickly by providing a wrapper around the core components that exposes an endpoint and an API.
- **UI:** The UI is for end-users, including the non-technical layman who wants to use PRIMEQA services interactively to ask questions and get answers.

3.3 Pipelines for OpenQA

PRIMEQA users can build an OpenQA *pipeline* and configure it to use any of the PRIMEQA retrievers and readers in a plug-and-play fashion. This is facilitated through a lightweight wrapper built around each core component, which implements the inference API (one of the PRIMEQA entry points). An example of such a pipeline can be connecting a ColBERT retriever to a generative reader based on LLMs such as those in the GPT series (Brown et al., 2020; Ouyang et al., 2022) or FLAN-T5 (Chung et al., 2022), providing retrieval-augmented generative QA capabilities. The retriever in this setting can provide relevant passages that can constitute part of the prompt for the LLM; this encourages answer generation grounded in those retrieved passages, reducing hallucination. Other pipelines can also be instantiated to use different retrievers (e.g., DPR, BM25) and readers (e.g., extractive, FiD) that are available through our model hub.

4 Services and Deployment

Industrial deployment often necessitates running complex models and processes at scale. We use Docker to package these components into micro-services that interact with each other and can be ported to servers with different hardware capabilities (e.g. GPUs, CPUs, memory). The use of Docker makes the addition, replacement or deletion of services easy and scalable. All components in the PRIMEQA repository are available via REST and/or gRPC micro-services. Our Docker containers are available on the public `DockerHub` and can be deployed using technologies such as OpenShift

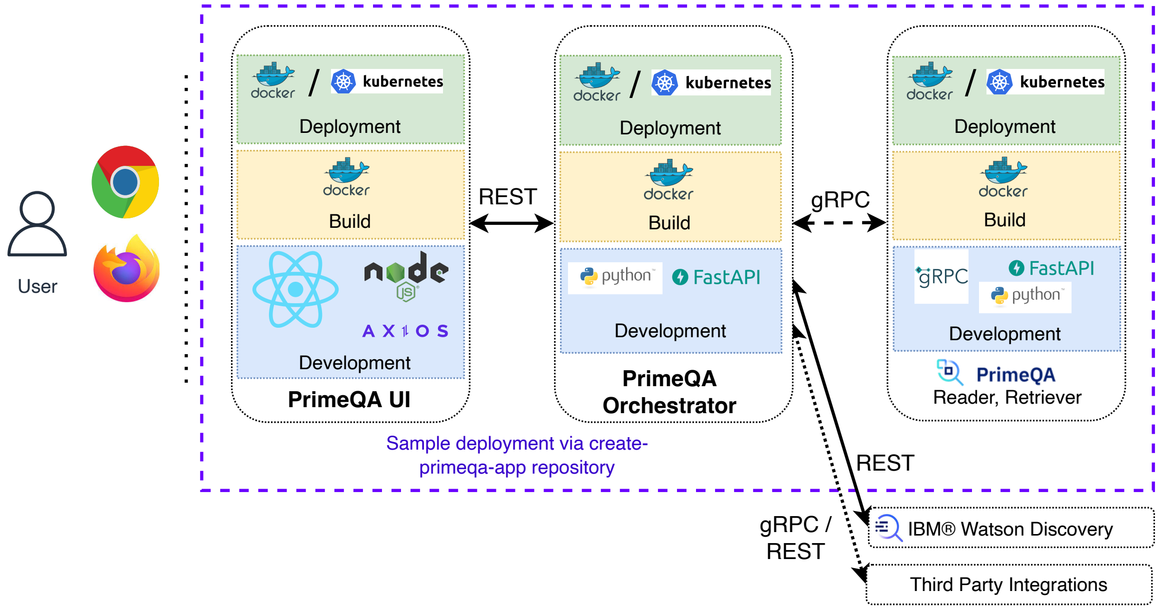

Figure 2: PRIMEQA's end-to-end application. Each container contains a development (blue), build (yellow) and deployment (green) stack.

and Kubernetes.

In addition to the main PrimeQA repository, we provide three sibling repositories for application deployment:

- primeqa-ui is the front-end UI. Users can personalize this by adding custom organization logos or changing display fonts.
- primeqa-orchestrator is a REST server and is the central hub for the integration of PRIMEQA services and external components and the execution of a pipeline. For instance, the orchestrator can be configured to search a document collection with either a retriever from PrimeQA such as ColBERT, or an external search engine such as Watson Discovery.[7]
- create-primeqa-app provides the scripts to launch the demo application by starting the orchestrator and UI services.

Figure 2 illustrates how to deploy a QA application at scale using the core PrimeQA services (e.g. Reader and Retriever) and our three sibling repositories. We provide this end-to-end deployment for our demo, however users can also utilize PrimeQA as an application with their own orchestrator or UI.

Figure 3 shows an OpenQA demo application built with the PRIMEQA components. In addition to providing answers with evidence, our demo application features a mechanism to collect user feedback. The *thumbs up / down* icons next to each result enables a user to record feedback which is then stored in a database. The user feedback data can be retrieved and used as additional training data to further improve a retriever and reader model.

5 Community Contributions

While being relatively new, PRIMEQA has already garnered positive attention from the QA community and is receiving constant successful contributions from both international academia and industry via Github pull requests. We describe some instances here and encourage further contributions from all in the community. We provide support for those interested in contributing through a dedicated slack channel [8], Github issues and PR reviews.

Neural Retrievers: ColBERT, one of our core neural retrievers, was contributed by Stanford NLP. Since PRIMEQA provides very easy entry points into its core library, they were able to integrate their software into the retriever script run_ir.py independently. Their contribution to PRIMEQA provides SOTA performance on OpenQA benchmarks by performing 'late interaction' search on a variety of datasets. They also contributed ColBERTv2 (Santhanam et al., 2022b) and its PLAID (Santhanam et al., 2022a) variant. The former reduces ColBERT index size by 10x while the latter makes search faster by almost 7x on GPUs.

[7] https://www.ibm.com/cloud/watson-discovery

[8] https://ibm.biz/pqa-slack

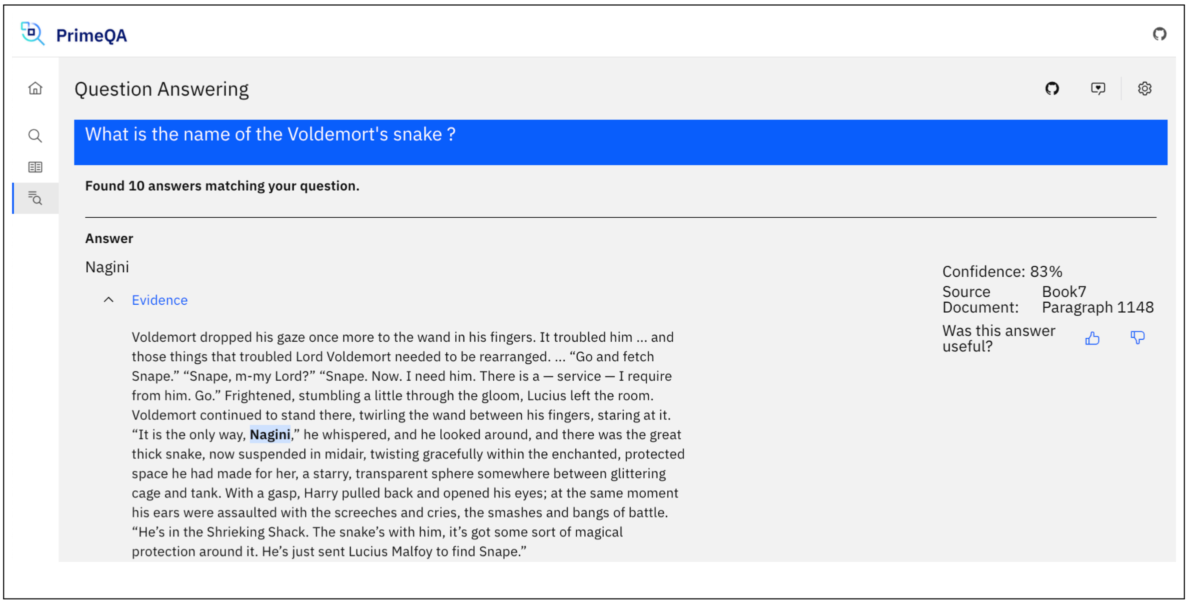

Figure 3: A custom OpenQA search application built with PRIMEQA. Additional screenshots are in Appendix A.

Few-shot Learning: The SunLab from Ohio State University added few-shot learning capabilities within PRIMEQA . Their contribution, ReasonBERT (Deng et al., 2021), provides a pre-trained methodology that augments language models with the ability to reason over long-range relations. Under the few-shot setting, ReasonBERT in PRIMEQA substantially outperforms RoBERTa (Liu et al., 2019)-based QA systems. PRIMEQA gives any researcher or developer the capability to easily integrate this component in their custom search application e.g. a DPR retriever and a ReasonBERT reader.

Table Readers: Beihang University and Microsoft Research Asia contributed Tapex (Liu et al., 2021) as the first generative Table reader in PRIMEQA. Tapex proposes a novel table pre-training strategy based on a neural SQL executor and achieves SOTA on Wiki-SQL (Zhong et al., 2017a) and Wiki-TableQuestions (Pasupat and Liang, 2015a). They utilize the Transformers (Wolf et al., 2020) sequence-to-sequence trainer for seamless integration into PRIMEQA. LTI CMU's NeuLab contributed OmniTab (Jiang et al., 2022b), which employs an efficient pre-training strategy leveraging both real and synthetic data. This integration happened organically as OmniTab builds on top of Tapex in PRIMEQA. Currently, their model yields the best few-shot performance on Wiki-TableQuestions, making it also an appropriate candidate system under domain shift.

Custom Search for Earth Science: NASA researchers created a custom search application for scientific abstracts and papers related to Earth Science which received global attention[9]. First, using the top level scripts in PRIMEQA, they trained an OpenQA system on over 100k abstracts by training a ColBERT retriever and an extractive reader. Then, they were able to quickly deploy the search application using the create-primeqa-app and make it available publicly[10].

6 Conclusion

PRIMEQA is an open-source Question Answering library designed by researchers and developers to easily facilitate reproduciblity and reusability of existing and future work in QA. This is an important contribution to the community, as it provides researchers and end users with easy access to state-of-the-art algorithms in the rapidly progressing field of QA. PRIMEQA also provides off-the-shelf models that developers can directly deploy for their custom QA applications. PRIMEQA is built on top of the largest open-source NLP libraries and tools, can incorporate LLMs through external APIs, and provides simple Python scripts as entry points for easy reuse of its core components. This straightforward access and high reusability has already garnered significant traction in the community, enabling PRIMEQA to grow organically as an important resource for rapid progress in QA.

[9] https://www.nextgov.com/emerging-tech/2023/02/ibm-nasa-will-use-ai-improve-climate-change-research/382437/

[10] http://primeqa.nasa-impact.net/qa

Ethics and Broader Impact

6.1 Broader Impact

PRIMEQA is developed as a one-stop and open-source QA repository with an aim to democratize QA research by facilitating easy replication and extension of state of the art methods in multilingual question answering and developments. QA is moving fast with the launch of state-of-the-art (SOTA) retrievers, readers and multi-modal QA models. However, there are two key hurdles which slow the adoption of the SOTA models by the community, which are (1) hard to reproduce for researchers and (2) involves a learning curve for developers to use in custom applications. PRIMEQA solves both the problems by providing multiple access points for different user groups for their easy adoption. PRIMEQA is licensed under Apache 2.0 and thus open to use in both academia and industry. Therefore, PRIMEQA can have an impact on the whole NLP community or more broadly any user working on NLP applications.

6.2 Ethical Considerations

The models available in PRIMEQA might inherit bias based on available training data in public domain. Such bias, if any, is in general present in the models contributed to PRIMEQA and not specific to PrimeQA. Therefore, the usage of PRIMEQA should be approached with the same caution as with any NLP model.

PRIMEQA supports easy access for researchers and developers to use state-of-the-art models and even customize them on their own data. However, PRIMEQA does not control the type of data the model will be exposed to in a custom environment. The general assumption is that these models will be used for rightful purposes in good faith.

Acknowledgments

We thank Lysandre Debut, Thomas Wolf and the Hugging Face team for their close collaboration with the PRIMEQA team to facilitate building PRIMEQA on top of the Hugging Face libraries and datasets. We also thank Chris Potts, Omar Khattab, Jon Saad-Falcon and Keshav Santhanam from Stanford University, Xiang Deng and Huan Sun from Ohio State University, Zhengbao Jiang from Carnegie Mellon University, Qian Liu from Beihang University, and Maximilian Schmidt from University of Stuttgart for contributing implementations of their state-of-the-art systems to PRIMEQA.

References

Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. 2019a. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, Florence, Italy. Association for Computational Linguistics.

Chris Alberti, Kenton Lee, and Michael Collins. 2019b. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*.

Akari Asai, Jungo Kasai, Jonathan Clark, Kenton Lee, Eunsol Choi, and Hannaneh Hajishirzi. 2021. XOR QA: Cross-lingual open-retrieval question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 547–564, Online. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *CoRR*, abs/2005.14165.

Rishav Chakravarti, Anthony Ferritto, Bhavani Iyer, Lin Pan, Radu Florian, Salim Roukos, and Avirup Sil. 2020. Towards building a robust industry-scale question answering system. In *Proceedings of the 28th International Conference on Computational Linguistics: Industry Track*, pages 90–101.

Saneem Chemmengath, Vishwajeet Kumar, Samarth Bharadwaj, Jaydeep Sen, Mustafa Canim, Soumen Chakrabarti, Alfio Gliozzo, and Karthik Sankaranarayanan. 2021. Topic transferable table question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4159–4172, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *Findings of EMNLP 2020*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonathan H. Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020a. TyDi QA: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Jonathan H Clark, Eunsol Choi, Michael Collins, Dan Garrette, Tom Kwiatkowski, Vitaly Nikolaev, and Jennimaria Palomaki. 2020b. Tydi qa: A benchmark for information-seeking question answering in typologically diverse languages. *Transactions of the Association for Computational Linguistics*, 8:454–470.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Deepset. 2021. Haystack.

Xiang Deng, Yu Su, Alyssa Lees, You Wu, Cong Yu, and Huan Sun. 2021. ReasonBERT: Pre-trained to reason with distant supervision. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6112–6127, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3558–3567, Florence, Italy. Association for Computational Linguistics.

Revanth Gangi Reddy, Bhavani Iyer, Md Arafat Sultan, Rong Zhang, Avirup Sil, Vittorio Castelli, Radu Florian, and Salim Roukos. 2021. Synthetic target domain supervision for open retrieval qa. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1793–1797.

Revanth Gangi Reddy, Vikas Yadav, Md Arafat Sultan, Martin Franz, Vittorio Castelli, Heng Ji, and Avirup Sil. 2022. Towards robust neural retrieval with source domain synthetic pre-finetuning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1065–1070, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Mueller, Francesco Piccinno, and Julian Eisenschlos. 2020a. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020b. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online. Association for Computational Linguistics.

IBM. 2020. Watson assistant.

Mohit Iyyer, Scott Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. In *Conference of the European Chapter of the Association for Computational Linguistics*.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022a. Omnitab: Pretraining with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942.

Zhengbao Jiang, Yi Mao, Pengcheng He, Graham Neubig, and Weizhu Chen. 2022b. OmniTab: Pre-training with natural and synthetic data for few-shot table-based question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 932–942, Seattle, United States. Association for Computational Linguistics.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. 2021. GooAQ: Open question answering with diverse answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 421–433, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vishwajeet Kumar, Saneem Chemmengath, Yash Gupta, Jaydeep Sen, Samarth Bharadwaj, and Soumen Chakrabarti. 2021. Multi-instance training for question answering across table and linked text.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019a. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019b. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019c. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Annual Meeting of the Association for Computational Linguistics*.

Yulong Li, Martin Franz, Md Arafat Sultan, Bhavani Iyer, Young-Suk Lee, and Avirup Sil. 2022. Learning cross-lingual IR from an English retriever. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4428–4436, Seattle, United States. Association for Computational Linguistics.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. In *International Conference on Learning Representations*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Shayne Longpre, Yi Lu, and Joachim Daiber. 2021. Mkqa: A linguistically diverse benchmark for multilingual open domain question answering. *Transactions of the Association for Computational Linguistics*, 9:1389–1406.

Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing,.

Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. 2021. Docvqa: A dataset for vqa on document images. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2200–2209.

Scott McCarley, Mihaela Bornea, Sara Rosenthal, Anthony Ferritto, Md Arafat Sultan, Avirup Sil, and Radu Florian. 2023. Gaama 2.0: An integrated system that answers boolean and extractive question. In *AAAI 2023*.

NVDIA. 2022. Tao toolkit.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. *arXiv preprint arXiv:1904.01038*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback.

Panupong Pasupat and Percy Liang. 2015a. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015b. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1470–1480, Beijing, China. Association for Computational Linguistics.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2022. Exploring the limits of transfer learning with a unified text-to-text transformer. 21(1).

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marcin-Elvis Guillem Andrew Robert, Ross and Thomas. 2022. Papers with code. In *Meta AI Research*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. 2022a. PLAID: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, CIKM '22, page 1747–1756, New York, NY, USA. Association for Computing Machinery.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022b. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. Association for Computational Linguistics.

Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. 2008. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge.

Siamak Shakeri, Noah Constant, Mihir Kale, and Linting Xue. 2021. Towards zero-shot multilingual synthetic question and answer generation for cross-lingual reading comprehension. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 35–45, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. 2020. On the importance of diversity in question generation for QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, Online. Association for Computational Linguistics.

Md Arafat Sultan, Avirup Sil, and Radu Florian. 2022. Not to Overfit or Underfit the Source Domains? An Empirical Study of Domain Generalization in Question Answering. In *EMNLP*.

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 *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Xiang Yue, Ziyu Yao, and Huan Sun. 2022. Synthetic question value estimation for domain adaptation of question answering. In *ACL*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017a. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017b. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103.

A Appendix

A.1 PrimeQA Applications

Figure 4 shows a screen-shot of three **PrimeQA** applications. Tables 2 and 3 provide lists of supported datasets and some important PRIMEQA links.

Datasets
OpenNQ
XOR-TyDi (Asai et al., 2021)
SQuAD (Rajpurkar et al., 2016)
TyDiQA (Clark et al., 2020b)
NQ (Kwiatkowski et al., 2019c)
ELI5
SQA (Iyyer et al., 2017)
WTQ (Pasupat and Liang, 2015b)
DocVQA (Mathew et al., 2021)
WikiSQL (Zhong et al., 2017b)

Table 2: A list of some of the supported datasets in PrimeQA

Retriever	Simple Python script
Reader	Inference APIs
Unstructured QG	Inference APIs
Pipeline	Inference APIs

Table 3: Links to PrimeQA

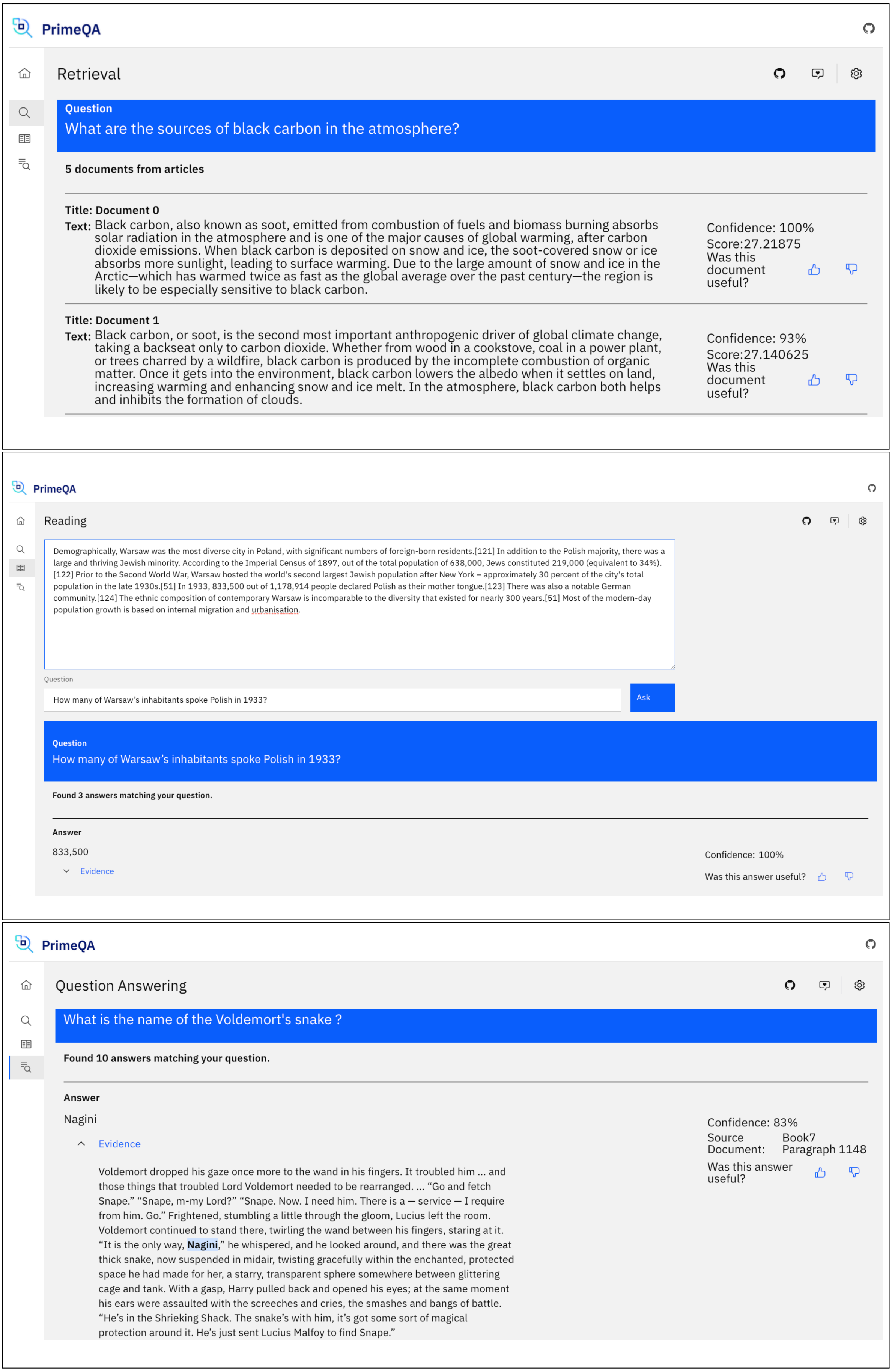

Figure 4: PrimeQA Applications