"Ask Me Anything": How Comcast Uses LLMs to Assist Agents in Real Time

Scott Rome
Tianwen Chen
scott_rome@comcast.com
tianwen_chen@comcast.com
Comcast AI Technologies
Philadelphia, PA, USA

Raphael Tang
raphael_tang@comcast.com
Comcast AI Technologies
Philadelphia, PA, USA

Luwei Zhou
Ferhan Ture
luwei_zhou@comcast.com
ferhan_ture@comcast.com
Comcast AI Technologies
Philadelphia, PA, USA

ABSTRACT

Customer service is how companies interface with their customers. It can contribute heavily towards the overall customer satisfaction. However, high-quality service can become expensive, creating an incentive to make it as cost efficient as possible and prompting most companies to utilize AI-powered assistants, or "chat bots". On the other hand, human-to-human interaction is still desired by customers, especially when it comes to complex scenarios such as disputes and sensitive topics like bill payment.[1]

This raises the bar for customer service agents. They need to accurately understand the customer's question or concern, identify a solution that is acceptable yet feasible (and within the company's policy), all while handling multiple conversations at once.

In this work, we introduce "Ask Me Anything" (AMA) as an add-on feature to an agent-facing customer service interface. AMA allows agents to ask questions to a large language model (LLM) on demand, as they are handling customer conversations—the LLM provides accurate responses in real-time, reducing the amount of context switching the agent needs. In our internal experiments, we find that agents using AMA versus a traditional search experience spend approximately 10% fewer seconds per conversation containing a search, translating to millions of dollars of savings annually. Agents that used the AMA feature provided positive feedback nearly 80% of the time, demonstrating its usefulness as an AI-assisted feature for customer care.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; **Language models**.

KEYWORDS

rag, llm, customer care, assistive AI, vector db, reranking

ACM Reference Format:
Scott Rome, Tianwen Chen, Raphael Tang, Luwei Zhou, and Ferhan Ture. 2024. "Ask Me Anything": How Comcast Uses LLMs to Assist Agents in Real Time. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '24), July 14–18, 2024, Washington, DC, USA*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3626772.3661345

[1]https://www.businessinsider.in/tech/enterprise/news/why-customers-still-prefer-speaking-to-a-human-not-chatbox/articleshow/91955173.cms

SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0431-4/24/07
https://doi.org/10.1145/3626772.3661345

1 INTRODUCTION

Comcast, like many other companies, provides customer service through various communication channels. Many self-service solutions are available on the mobile "Xfinity" app (e.g., reviewing latest bill) which also has an option to chat with an AI-powered bot named "Xfinity Assistant". While these digital automation capabilities have been replacing human customer representatives (also referred to as "agents") for many tasks, there are still many situations that require human-to-human interactions.

A customer trying to simply look up information about their profile, internet services, or bill, they should be able to do it without an agent's assistance. This also holds true if they are trying to carry out a relatively straightforward task like rescheduling their appointment or make a change to their services.

Past studies show a human-human interaction is preferred over a human-computer one in certain customer service situations[21]. For example, agents might outperform bots in situations that require creative problem solving. In other situations, the customer might simply prefer to talk to a agent to benefit from their empathy and emotional intelligence, or to navigate through cultural sensitivities.

At Comcast, an internal custom tool suite aims to help agents to effectively and efficiently handle such conversations. However, it still often requires manually looking up information in multiple places, relating it to what the customer is saying, then crafting a relevant response that aligns with the communication guidelines. In this paper, we introduce a new feature to this tool suite called "Ask Me Anything" (AMA). It leverages large language models (LLMs) following a retrieval-augmented generation (RAG) approach to generate contextually relevant responses by combining internal knowledge sources, indexing existing knowledge articles efficiently at build time, retrieving relevant chunks of text for a given question at query time, then feeding them to a *Reader* LLM to generate a succinct answer with citations provided as reference. In the next section, we describe the methodology in more detail.

2 METHODOLOGY

Our system follows a typical RAG implementation with modifications to improve performance on proprietary questions. First, the documents are preprocessed to text and chunked, the chunks are embedded then stored with metadata (e.g., associated URL for citations, an identifier, the title, etc.) in a vector database. We describe our specific choices for processing and embeddings in Section 2.1 and Section 2.2 respectively with some experimental justification. Next, we detail how we train and evaluate a reranking model using synthetic data to improve search result relevancy in Section 2.3. Finally, we discuss how we generate answers followed by how we evaluate the system in Section 2.4 and 2.5.

2.1 Document Preprocessing

We receive documents from various internal clients in different formats. We standardize the documents into plain text and chunk each document into snippets using Deepset.ai's Haystack library [13]. In order to uniquely reference each chunk of every document after retrieval, we assign an origin identifier to each document and a local identifier to each chunk. Finally, we implement role-based access control on each document, so different users can only view the documents for which they have permission.

In Table 1, we show various chunking parameters for Haystack's preprocessor and their evaluation scores. The metric derivation is explained in Section 2.5 (Answer quality assumed the top 3 items were passed to the LLM). We observed a large improvement from setting a higher `max_chars_check`, which we used as a proxy for limiting the size of each snippet given to the LLM.

Table 1: Chunking parameters and evaluation of three different settings.

Parameter	A	B	C
clean_empty_lines	true		
clean_whitespace	true		
clean_header_footer	true		
split_by	word		
split_length	300	100	
split_overlap	50	25	
split_respect_sentence_boundary	true		
max_chars_check	1000		3000
Metric			
Answer Quality	-	-5.7%	+13.2%
MRR	-	-13.3%	0.0%
R@3	-	-7.9%	0.0%
NDCG	-	-10.0%	0.0%

For clarity, only changes from setting A are found in the table. Empty parameter values mean they are same as A. The metric values are the relative difference from A, i.e., $100 \cdot (\mu_B - \mu_A)/\mu_A$ for some metric μ. Metrics are defined in Section 2.5.

2.2 Retrieving Relevant Text Snippets

To inform the choice of our retriever model, we conducted pilot experiments on a curated evaluation set of fifty question–answer pairs. We searched the in-production system logs for queries starting with a WH-word (who, what, how, etc.) or ending with a question mark, roughly following the procedure on Bing query logs from WikiQA [24]. For each question, we then located the relevant passage and answer span in our internal knowledge base used by agents. Queries without answers were also labeled as such. Crucially, this process avoids back-formulation [17], where queries are manually written by annotators based on known passages rather than crawled from logs, resulting in biased evaluation sets.

We experimented with both dense and sparse retrieval models. For the sparse model, we used Okapi BM25 [16] with $k_1 = 1.0$ and $b = 0.5$. For the dense ones, we experimented with four: dense passage retrieval (DPR) [9], fine-tuned on Natural Questions [10]; MPNet-base (v1) [18], trained on 160GB of text corpora including Wikipedia, BookCorpus [26], and OpenWebText [6]; OpenAI's state-of-the-art `ada-002` embeddings model; and MPNet-base v2, trained further on one billion sentence pairs for better embedding quality.[2] Each was deemed to satisfy our computational and financial constraints at inference time.

In Table 2, we report the recall@3 (R@3) and the mean reciprocal rank (MRR) of these models on our evaluation set. The choice of recall@3 (versus recall@5 or 10) is from us feeding the top-three retrieved passages into the LLM. As a sanity check, we also ran a baseline that randomly drew a passage, which unsurprisingly yielded low scores. Mirroring prior work [23], we found that BM25 remains a strong baseline, outperforming DPR in R@3 and MRR, respectively. We conjecture that this results from Natural Questions being substantially out of domain from our data.

Table 2: Results of various retrievers on our pilot evaluation set

Method	Recall@3	MRR
Random	-71.4%	-83.9%
BM25	-	-
DPR (`single-nq`)	-42.8%	-42.9%
DPR (`multiset-nq`)	-23.8%	-29.0%
Multi-QA MPNet-base	+33.0%	+39.7%
OpenAI embeddings (`ada-002`)	+33.0%	+53.9%
MPNet-base v2	**+38.1%**	**+54.9%**

Statistics presented as relative difference from BM25, i.e., $100 \cdot (\mu - \mu_{BM25})/\mu_{BM25}$. Underline denotes statistical significance relative to DPR.

We observe MPNet-base (v1), OpenAI's `ada-002`, and MPNet-base (v2) to perform similarly. Signed-rank tests for R@3 and t-tests for MRR also reveal a significant difference ($p < 0.05$) from DPR. Due to operational convenience and the high performance of OpenAI's ADA embeddings, we used ADA for the retriever component for the final system.

For our production retrieval step, we embedded both the title of the article and the text of the individual chunk and added them together prior to storage in the vector database. Anecdotally, we found this to yield a more comprehensive retrieval for a variety of queries, especially when chunks were missing some descriptive context of the topic of the article.

[2]Nils Reimers's open-source contribution: https://discuss.huggingface.co/t/train-the-best-sentence-embedding-model-ever-with-1b-training-pairs/7354

Table 3: Training hyperparameters.

Parameter	Specification
Learning Rate	5×10^{-6}
Batch Size	8
Number of GPUs[1]	10
Warmup Steps	4000
Weight Decay	0.001
Epochs	1
Total Training Steps	171391
Learning Rate Scheduler	Warmup-constant

[1] GPU type: `g4dn.xlarge` (Nvidia T4)

2.3 Reranking Search Results

We found that reranking results using models finetuned on synthetic data improved the retrieval step. Our approach was inspired by previous synthetic data generation approaches [1, 3]. First, we used GPT-4 to generate synthetic questions from each snippet in our dataset. We then ran each question through our search system using OpenAI's `text-embeddings-ada-002` [8] embeddings. Any questions where the original snippet used for question generation did not appear in the top 20 results were discarded. For each synthetic question, we stored the top 20 items retrieved, their relevance as scored by `BGE-reranker-large` [22], and an indicator that the snippet was the source of the question. The final rankings were determined by first placing the source snippet as the "most relevant" result, followed by the snippets in most relevant order as scored by the `BGE-reranker-large` model.

For training, we used RankNet [2] to distill these rankings into a finetuned MPNet [18], in particular `all-mpnet-base-v2` from `sentence-transformer` [15], which has fewer parameters requiring less computational resources to deploy into production than `BGE-reranker-large`. The final dataset after constructing the necessary pairs for RankNet consisted of over 10 million examples. We set aside 0.5% of the examples as validation dataset. Our training parameters were listed in Table 3. We used `DistributedDataParallel` from PyTorch [12] for distributed training, so the effective batch size is the number of GPUs multiplied by the batch size. We found the "Linear Scaling Rule", where one scales the learning rate when the batch size increases, to not apply to our use case [7], but we suspect it is because the original MPNet architecture was trained with a much larger batch size than we used for finetuning.

To further evaluate the performance of our reranker model, we randomly sampled 10,000 real questions asked by customer service agents in our production system. For every retrieved document, we followed the approach in [20], which showed that an LLM can accurately predict the relevancy of search results. Specifically, GPT-4 was used to evaluate the overall quality of each document to the question, which combined the scores from how the document matches the intent of the question as well as how trustworthy the document is. The final integer score ranged between 0 and 2, with higher score meaning higher overall quality. Table 4 compares multiple metrics between ADA vs. reranker. Since the overall score is non-binary, we compute MRR using the rank of first document with a score of 2, and recall@3 examines whether the top 3 documents contain any documents with a score of 2. The results indicate an improvement in retrieval performance with the reranker model.

Table 4: ADA vs. Reranker Search Results using Production Questions

Metric	ADA	Reranker
Recall@3	-	+12%
MRR	-	+15%
NDCG	-	+4.8%

For clarity, only changes from setting ADA are found in the table. The metric values are the relative difference from ADA.

2.4 Generating the Answer from Snippets

In generating the answer, we follow the conventional wisdom approach in the RAG literature. We begin our prompt with a preamble of guidelines for the model, followed by the task description. Due to the length of our snippets of text from the knowledge base, we are unable to provide few-shot examples. We have anecdotally found it better to include more of the text to avoid necessary information being cut off at random. To avoid the "lost in the middle" problem [11], we reverse the order of the Top K results when passed into the LLM, formatted as XML capturing the ID, title and content of the result. We used OpenAI's `gpt-3.5-turbo` for our production Reader component. As a final step in our prompt, we ask the LLM to answer the given question using the search results.

2.4.1 *Citations.*

An important product feature of of the AMA solution is providing references to agents so they can learn more about the answer given. This can be seen in various RAG implementations, such as Microsoft Copilot. In addition, the goal was to build confidence in the system's output and drive adoption internally. Inspired by the Fact-Checking Rail [14], our Citation Rail was accomplished by prompting the LLM to cite its sources in a specific manner (c.f., Figure 1) combined with a post processing step where the citations were removed from the text. If no citations were found, then the system would not return the answer. Practically, there was another benefit from an *observability* perspective: through this approach, we identified most "no answer" responses from the LLM, as typically the LLM would response similarly to "I'm sorry. I was unable to find the answer in the documents" without a citation.

```
Please include a single source at the end of
     your answer, i.e., [Document0] if
    Document0 is the source. If there is
    more than one source, use [Document0][
    Document1] if Document0 and Document1
    are the sources.
```

Figure 1: An example component of a prompt to encourage citations from the LLM used in the system prompt section.

Table 5: Response Quality

Metric	ADA	Reranker
Answer Quality	-	+5.9%
Citation Match Rate	-	+2.5%
Recall@3	-	+16.5%

For clarity, only changes from setting ADA are found in the table. The metric values are the relative difference from ADA.

2.5 Offline Response Evaluation

To evaluate the system's responses, we follow the LLM-as-a-judge methodology [25], in addition to metrics around retrieval quality typical of a search system. In particular, a random sample of questions from customers were pulled from production traffic. Human annotators then wrote correct answers to each query using internal knowledge bases that are also available to the AMA system. We were able to compare system answers to correct responses given by human annotators using GPT-4 to compute "Answer Quality".

For each question, the annotators also provided a citation from which their answers were based. We used this to calculate "Citation Match Rate": the percentage of cases in which the citation from the AMA system matched the ground truth. Given that our retrieval step returns a list, we calculated Recall@K by assuming the annotated citation is the only relevant document.

Table 5 shows key metrics for the same two approaches as in Table 4 (`text-embedding-ada-002` for dense retrieval of relevant documents and rerankering the ADA-retrieved documents using our finetuned model). We observe that using reranked documents, LLM is able to achieve a higher answer quality meaning that the answer from a different document ranking is more accurate according to GPT-4. The improvement can also be explained by the increased Citation Match Rate and Recall@3 from the reranked documents directly influencing the LLM's ability to answer accurately.

3 DEPLOYING AMA TO CUSTOMER SERVICE AGENTS

Due to business sensitivity purposes, this section will obscure some details related to monetary business metrics. The system was piloted with hundreds of chat agents in late 2023. Over the course of a month-long trial, chat handling time improved 10% when agents used AMA versus the traditional search option, which required the agent to open a new tool and perform a search. We believe this is a good proxy metric for answer quality because an inaccurate or incomplete response from AMA would require the agent to start over and revert to the traditional option, duplicating work and taking more time overall. Explicit feedback, via a simple thumbs up/thumbs down UI element, was also collected from agents, with nearly an 80% positive feedback rate (there is no baseline for this rate as such feedback was not requested before the release of this feature). Shortly after the trial period, the system was rolled out to all chat agents (in thousands), with AMA-driven search becoming the preferred way of searching, accounting for two thirds of all typed queries.

4 ONLINE RERANKER EXPERIMENT

Shortly after the trial from Section 3 concluded, we began an A/B test of the reranker module described in Section 2.3. The control variant used only the ADA embeddings for vector retrieval with no reranking component, and the treatment utilized the reranker component on the top 20 results from the ADA-based vector retrieval step. The test ran for three weeks in early 2024. We powered our tests at 80% and use significance level $\alpha = .01$ for metrics that applied to every interaction and $\alpha = .05$ when metrics considered user feedback, as responses were sparse. Due to the limited pool of agents, our randomization unit, we utilized an agent-day randomization similar to the cookie-day randomization found in other large systems [19] to increase statistical power. It has been shown in the literature [5] that violations of the independent and identically distributed (IID) assumption can lead to underestimation of the variance, but these tests can still be considered trustworthy in practice by using smaller significance thresholds and when observing larger effect sizes. The delta method [4] was employed to estimate the variance from question-level metrics.

We observed a statistically significant increase in two of our metrics: namely the "No Answer Rate", which is the number of queries with no answer divided by the total number of queries, and the "Positive Feedback Rate", defined as the number of thumbs up divided by the count of feedback received. Downstream business metrics like average handle time and escalation rate showed no significant difference. However, the improvement in No Answer Rate implies that the system was able to handle more questions than before by providing the relevant documents to the LLM while also increasing the rate of positive feedback.

Table 6: A/B Test Results

Metric	Effect	p-value
No Answer Rate	-11.9%	$p < .001$
Positive Feedback Rate	+8.9%	$p < .05$

Table contains relative change from control as the effect. Lower is better for No Answer Rate.

5 CONCLUSIONS

In this paper, we introduced AMA, a large-scale solution to a common business need: efficient high-quality customer care. Through the use of third-party LLMs and proven RAG methodology, we were able to build AMA pretty quickly and demonstrate clear value as an assistive feature. We showed improvements to retrieval and answer quality with specific choices for the document preprocessing, the retrieval model and its embeddings, as well as a custom reranker model. As we deploy AMA to thousands of agents with tangible business benefits, we believe that this provides a good example of how humans and AI can collaborate to better serve customers.

REFERENCES

[1] Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. InPars: Data Augmentation for Information Retrieval using Large Language Models. arXiv:2202.05144 [cs.CL]

[2] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning* (Bonn, Germany) *(ICML '05)*. Association for Computing Machinery, New York, NY, USA, 89–96. https://doi.org/10.1145/1102351.1102363

[3] Zhuyun Dai, Vincent Y. Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B. Hall, and Ming-Wei Chang. 2022. Promptagator: Few-shot Dense Retrieval From 8 Examples. arXiv:2209.11755 [cs.CL]

[4] Alex Deng, Ulf Knoblich, and Jiannan Lu. 2018. Applying the Delta Method in Metric Analytics: A Practical Guide with Novel Ideas. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (London, United Kingdom) *(KDD '18)*. Association for Computing Machinery, New York, NY, USA, 233–242. https://doi.org/10.1145/3219819.3219919

[5] Alex Deng, Jiannan Lu, and Jonthan Litz. 2017. Trustworthy Analysis of Online A/B Tests: Pitfalls, challenges and solutions. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 641–649. https://doi.org/10.1145/3018661.3018677

[6] Aaron Gokaslan and Vanya Cohen. 2019. OpenWebText Corpus. http://Skylion007.github.io/OpenWebTextCorpus.

[7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2018. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. arXiv:1706.02677 [cs.CV]

[8] Ryan Greene, Ted Sanders, Lilian Weng, and Arvind Neelakantan. 2022. *New and improved embedding model*. https://openai.com/blog/new-and-improved-embedding-model

[9] 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)*. 6769–6781.

[10] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[11] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172 [cs.CL]

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. (2017).

[13] Malte Pietsch, Timo Möller, Bogdan Kostic, Julian Risch, Massimiliano Pippi, Mayank Jobanputra, Sara Zanzottera, Silvano Cerza, Vladimir Blagojevic, Thomas Stadelmann, Tanay Soni, and Sebastian Lee. 2019. Haystack: the end-to-end NLP framework for pragmatic builders. https://github.com/deepset-ai/haystack.

[14] Traian Rebedea, Razvan Dinu, Makesh Sreedhar, Christopher Parisien, and Jonathan Cohen. 2023. NeMo Guardrails: A Toolkit for Controllable and Safe LLM Applications with Programmable Rails. arXiv:2310.10501 [cs.CL]

[15] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. http://arxiv.org/abs/1908.10084

[16] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.

[17] Tetsuya Sakai, Yoshimi Saito, Yumi Ichimura, Tomoharu Kokubu, and Makoto Koyama. 2004. The effect of back-formulating questions in question answering evaluation. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*. 474–475.

[18] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. arXiv:2004.09297 [cs.CL]

[19] Diane Tang, Ashish Agarwal, Deirdre O'Brien, and Mike Meyer. 2010. Overlapping Experiment Infrastructure: More, Better, Faster Experimentation. In *Proceedings 16th Conference on Knowledge Discovery and Data Mining*. Washington, DC, 17–26.

[20] Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2023. Large language models can accurately predict searcher preferences. arXiv:2309.10621 [cs.IR]

[21] Kaitlin Wowak. [n. d.]. Humans vs. automation: Service center agents can outperform technology, study shows. https://news.nd.edu/news/humans-vs-automation-service-center-agents-can-outperform-technology-study-shows/

[22] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-Pack: Packaged Resources To Advance General Chinese Embedding. arXiv:2309.07597 [cs.CL]

[23] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy Lin. 2019. Critically examining the "neural hype": weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*. 1129–1132.

[24] Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2013–2018.

[25] Lianghui Zhu, Xinggang Wang, and Xinlong Wang. 2023. JudgeLM: Fine-tuned Large Language Models are Scalable Judges. arXiv:2310.17631 [cs.CL]

[26] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*. 19–27.