Sustainable Extractive Question Answering for Resource-Constrained Personalized Document Assistants

Sayali Nitin Doifode[1,*]

[1]Department of Informatics, King's College London, London, United Kingdom

Abstract

Large Language Models (LLMs) have become the dominant paradigm for personalized question answering and conversational assistants. However, their deployment demands significant computational resources, raising sustainability and accessibility concerns. This is particularly relevant in resource-constrained settings such as classrooms, healthcare training environments, and low- and middle-income countries (LMICs). This paper presents a fully offline, zero-dependency extractive question answering system designed for deployment on standard laptops without GPU or internet access. The proposed system combines a pure-Python TF-IDF retriever with a multi-factor extractive answer selector to deliver document-grounded answers with sub-millisecond latency and under 10 MB of memory. We frame the system as a sustainable first-pass component in cascaded QA architectures, where high-confidence queries are handled extractively and only low-confidence queries are routed to costlier LLM-based systems. Evaluated on the full SQuAD v2.0 development set (11,873 questions across 35 articles), the system achieves 69.0% answer containment and 0.171 token F1 on answerable questions at a mean latency of 0.097 ms. Back-of-envelope energy estimates indicate the extractive approach consumes approximately 0.0015 mJ per query versus 120–675 mJ for quantized small language models. This represents a reduction of over four orders of magnitude. Ablation studies over scoring weights, chunk sizes, and refusal thresholds demonstrate the tunability of the pipeline. We discuss how the modular architecture supports personalisation through user-specific document collections and adaptive confidence thresholds, positioning the system as a sustainable complement to LLM-based personalisation pipelines. The code is publicly available at https://github.com/sayali1901/sustainable-extractive-qa.

Keywords

sustainable AI, resource-constrained QA, extractive question answering, offline personalisation, TF-IDF retrieval, green NLP, cascaded QA architectures

1. Introduction

The rapid adoption of Large Language Models (LLMs) for personalized question answering and document assistance has produced remarkable advances in accuracy and fluency [1, 2]. Retrieval-Augmented Generation (RAG) pipelines combine external document retrieval with generative models. They have become the standard architecture for grounded QA systems [3]. The first LLM4Good workshop at UMAP 2025 highlighted these opportunities. The contributions explored LLM-based educational guidance [4, 5] and responsible AI deployment [6]. Recent surveys have further mapped the landscape of LLM personalisation [7, 8].

However, deploying LLM-based systems entails substantial computational costs. Even quantized small language models (SLMs) such as TinyLlama require hundreds of megabytes of RAM and seconds of inference time per query on consumer hardware [9]. Moreover, larger models demand dedicated GPUs and cloud infrastructure. Faiz et al. [10] developed LLMCarbon to model the end-to-end carbon footprint of such systems. This revealed that inference-time emissions can dominate the lifecycle cost when models serve many users. Strubell et al. [11] demonstrated that training a single large NLP model can emit as much carbon as five automobiles. Also, Schwartz et al. [12] advocated for *Green AI*, which is research that prioritises computational efficiency as well as accuracy.

Beyond environmental impact, resource demands create accessibility barriers. Educators in under-funded schools, healthcare trainees in rural clinics, and learners in LMICs often lack the hardware,

Joint Proceedings of the ACM UMAP Workshops 2026, UMAP 2026, June 8–11, 2026, Gothenburg, Sweden

*Corresponding author.

sayalinitin02@gmail.com (S. N. Doifode)

connectivity, or budget to deploy LLM-based assistants. If personalised document assistance is only available to users with powerful hardware and reliable internet, then the technology reinforces rather than reduces informational inequality.

This paper introduces a fully offline, zero-dependency extractive QA system designed to operate within extreme resource constraints. Rather than positioning the system as a replacement for LLM-based personalisation, we frame it as a *sustainable first-pass filter* in cascaded QA architectures. High-confidence factoid queries are handled extractively at negligible cost. Only low-confidence or complex queries are routed to more resource-intensive generative models. Our key contributions are:

1. A complete extractive QA pipeline implemented in pure Python with zero external dependencies. The code is publicly available.[1]
2. An evaluation on the full SQuAD v2.0 development set (11,873 questions) demonstrating 69.0% answer containment and 0.171 token F1 at sub-millisecond latency.
3. Energy estimates showing the extractive approach consumes approximately four orders of magnitude less energy per query than quantized SLMs.
4. Ablation studies over scoring weights, refusal thresholds, and chunk sizes, including a finding that coverage-only scoring outperforms the multi-factor default at scale.
5. A discussion of how the modular architecture supports personalisation extensions relevant to the UMAP community.

2. Related Work

Retrieval-Augmented QA. The retrieve-then-read paradigm was popularised by Chen et al. [13], who combined TF-IDF retrieval with neural reading comprehension. Dense passage retrieval [14] later improved recall using learned embeddings, and RAG [3] integrated retrieval into generation end-to-end. While accuracy has improved, computational requirements have grown correspondingly. Our work revisits the classical TF-IDF approach and evaluates how far it can go on a standard QA benchmark without any external dependencies.

Efficient and Small Language Models. Efforts to reduce model size include knowledge distillation [15], quantization [16], and purpose-built small models [9]. Xu et al. [17] survey efficient inference techniques. Despite these advances, even the smallest viable SLMs require hundreds of megabytes and seconds of latency per query on CPU hardware.

Sustainable NLP and the LLM4Good Community. Strubell et al. [11] first quantified the environmental cost of large model training. Schwartz et al. [12] proposed reporting efficiency metrics alongside accuracy. Faiz et al. [10] extended this to inference-time carbon modelling. Liang et al. [18] advocate holistic evaluation including computational cost. The inaugural LLM4Good workshop at UMAP 2025 brought together researchers addressing these challenges in personalisation contexts, with papers on LLM-based educational recommenders [5] and formative feedback systems [4]. Our work extends this perspective by demonstrating that for many document QA tasks, the marginal accuracy gain from LLMs does not justify the resource overhead.

Personalisation in Document QA. Personalised QA systems adapt their behaviour based on user context, including document collections, interaction history, and stated preferences [7, 8]. The LaMP benchmark [8] evaluates LLM personalisation across tasks requiring user-specific context. While most personalisation research focuses on generative models, the underlying principle, that is, adapting system behaviour to individual users, can also be applied to extractive pipelines through user-specific corpora and adaptive scoring parameters. We discuss these extensions in Section 5.

[1] https://github.com/sayali1901/sustainable-extractive-qa

Documents → C1: Chunker → C2: TF-IDF Retriever → C3: Extractive QA → Answer + Citations

C4: Audit Logger ← (logs every stage)

Figure 1: System pipeline overview. Each component is independently replaceable. The modular design supports personalisation through user-specific document collections (C1), adaptive retrieval parameters (C2), and tunable confidence thresholds (C3).

3. System Architecture

The proposed system comprises four modular components operating in a sequential pipeline (Figure 1), implemented in approximately 900 lines of Python with no external dependencies.

3.1. C1: Document Chunker

Documents are segmented using a fixed-window strategy with window size W and overlap O (default: $W = 200$, $O = 40$ tokens). Each chunk receives a deterministic MD5 identifier from its content and positional index, ensuring reproducibility. Page and section metadata are preserved for citation. In a personalised deployment, each user maintains their own document collection (e.g., lecture notes, textbook chapters), and the system indexes each collection independently.

3.2. C2: TF-IDF Retriever

The retriever implements TF-IDF scoring from scratch in pure Python. For term t in chunk d within a corpus of N chunks:

$$\text{tfidf}(t, d) = \frac{f_{t,d}}{|d|} \times \left(\log \frac{N+1}{n_t + 1} + 1 \right) \tag{1}$$

where $f_{t,d}$ is the raw count, $|d|$ is total tokens, and n_t is chunk frequency. Smoothed IDF prevents division-by-zero. Retrieval ranks by cosine similarity, returning top-$k = 5$ chunks.

3.3. C3: Extractive QA Engine

Rather than generating answers, the system extracts sentences from retrieved chunks using a multi-factor scoring function. For candidate sentence s with query tokens Q and sentence tokens T_s:

$$\text{score}(s) = \alpha \cdot \frac{|T_s \cap Q|}{|Q|} + \beta \cdot \frac{|T_s \cap Q|}{|T_s|} + \gamma \cdot \min\left(\frac{|T_s|}{30}, 1 \right) \tag{2}$$

with default weights $\alpha = 0.5$ (coverage), $\beta = 0.3$ (density), $\gamma = 0.2$ (length). The final score incorporates retrieval ranking and diversity:

$$\text{final}(s) = 0.6 \cdot \text{score}(s) + 0.4 \cdot \text{ret\_score}(\text{chunk}(s)) + \epsilon_{\text{div}} \tag{3}$$

where $\epsilon_{\text{div}} = 0.1$ if the chunk has not yet contributed a citation.

Refusal mechanism. If $\max_s \text{final}(s) < \theta$, the system refuses: "Insufficient evidence." When a latency budget is exceeded, retrieval-only results are returned (graceful degradation). The pseudocode is shown in Algorithm 1.

3.4. C4: Audit and Determinism

Every run produces a structured JSON audit log recording run ID, timestamp, model and prompt versions, query, latency, confidence, citation count, and determinism flag. Fixed seeds ($s = 42$) and pure Python arithmetic ensure bitwise reproducibility. When the system is deployed with user-specific document collections, the log records which documents contributed to each answer, enabling users to verify answer provenance.

Algorithm 1 Extractive QA with Refusal

Require: query q, chunks $\mathcal{C}$, threshold θ

$\mathbf{q} \leftarrow \text{tfidf\_vector}(q)$
top_chunks $\leftarrow$ top-$k(\cos(\mathbf{q}, \mathcal{C}))$
for each sentence s in top_chunks **do**
  $\text{score}(s) \leftarrow \alpha \cdot \text{cov}(s, q) + \beta \cdot \text{den}(s, q) + \gamma \cdot \text{len}(s)$
  $\text{final}(s) \leftarrow 0.6 \cdot \text{score}(s) + 0.4 \cdot \text{ret}(s) + \epsilon_{\text{div}}$
end for
if $\max_s \text{final}(s) < \theta$ **then**
  return "Insufficient evidence"
else
  return top-scoring sentence with chunk citations
end if

4. Experimental Evaluation

4.1. Dataset

We evaluate on the full SQuAD v2.0 development set [19], which contains 11,873 questions across 35 Wikipedia articles. Of these, 5,928 are answerable and 5,945 are unanswerable. Each question is associated with a specific paragraph context. We use paragraph-level evaluation: each question is answered using its associated paragraph as the document, matching the use case of a personal document assistant where a user has a specific document and asks questions about it.

4.2. Metrics

We report the following metrics:

- **Answer Containment:** Whether the extracted sentence contains the gold answer span. This is the primary utility metric. Because the system returns full context sentences rather than minimal spans, a user receives the correct answer embedded within a readable sentence.
- **Token F1:** The harmonic mean of token-level precision and recall between the extracted text and the gold answer. Because the system returns full sentences, token F1 is lower than for span-extraction systems by design. We report it for comparability with published SLM benchmarks.
- **Exact Match (EM):** Strict string match between extracted text and gold answer. For a sentence-level extractive system, EM is expected to be near zero because the system returns full sentences, not minimal answer spans. We report it for completeness but it does not reflect practical utility.
- **Refusal Accuracy:** The proportion of unanswerable questions correctly refused.
- **p95 Latency:** The 95th percentile query latency in milliseconds.

4.3. Results

Table 1 summarises the system's performance. The system achieves 69.0% answer containment on answerable questions. This means that in roughly 7 out of 10 cases, the extracted sentence contains the correct answer. The token F1 of 0.171 is lower than span-extraction systems because the system returns full context sentences, not minimal spans.

Exact match is near zero (0.19%). This is not a failure of the system but a consequence of the extractive design. SQuAD gold answers are short spans (e.g., "1066"), and the system returns full sentences (e.g., "The Norman conquest of England began in 1066."). The answer is correct and contained, but EM does not capture this. Answer containment is the appropriate metric for evaluating sentence-level extraction.

Table 1
Evaluation results on the full SQuAD v2.0 development set (11,873 questions, paragraph-level context).

Metric	Value
Total questions	11,873
Answerable	5,928
Unanswerable	5,945
Answer Containment	69.0%
Average Token F1	0.171
Exact Match	0.19%
Refusal Accuracy ($\theta = 0$)	0.0%
Mean Latency	0.097 ms
p95 Latency	0.183 ms
Peak Memory	$<$10 MB
Est. Energy per Query	$\sim$0.0015 mJ
Deterministic	✓

Table 2
Ablation over scoring weights (α, β, γ). Evaluated on 5,928 answerable questions.

Configuration (α, β, γ)	Contain.	F1
Default (0.5, 0.3, 0.2)	69.0%	0.171
Coverage-only (1.0, 0.0, 0.0)	75.0%	0.187
No length (0.6, 0.4, 0.0)	70.6%	0.183
Equal (0.33, 0.33, 0.33)	63.7%	0.156
Density-heavy (0.2, 0.6, 0.2)	60.8%	0.159

Table 3
Refusal threshold θ sensitivity. Higher θ catches more unanswerable questions but reduces containment on answerable ones.

θ	Contain.	F1	Refusal
0.00	69.0%	0.171	0.0%
0.25	68.2%	0.169	3.8%
0.30	66.0%	0.163	10.2%
0.35	61.4%	0.149	20.9%
0.40	54.5%	0.131	36.0%
0.45	44.7%	0.105	52.9%
0.50	33.7%	0.078	68.3%

4.4. Ablation Studies

Scoring weight sensitivity. Table 2 shows the effect of varying the coverage/density/length weights (α, β, γ) on answer containment. An interesting finding is that coverage-only scoring ($\alpha = 1.0$) outperforms the multi-factor default (75.0% vs 69.0%). This suggests that at scale, query term coverage is the most important factor and the density and length components add complexity without benefit. However, coverage-only scoring may overfit to certain question types. The multi-factor design still provides more balanced scoring for edge cases.

Refusal threshold sensitivity. Table 3 shows the trade-off between refusal accuracy and answer containment as θ varies.

At $\theta = 0.35$, the system correctly refuses 20.9% of unanswerable questions while maintaining 61.4% containment. At $\theta = 0.50$, refusal accuracy reaches 68.3% but containment drops to 33.7%. The full trade-off curve enables deployment-specific tuning. In a cascaded architecture, this threshold controls which queries stay on the fast extractive path and which get routed to an SLM.

Chunk size sensitivity. Table 4 shows the effect of varying window size W with overlap $O = W/5$.

Answer containment is stable across all chunk sizes (67.7%–69.3%). This is a useful result for deployment: the pipeline is robust to this parameter, so users and developers do not need to tune it

Table 4
Chunk size sensitivity on 5,928 answerable questions.

W	O	**Contain.**	**F1**	**p95 Lat.**
50	10	67.7%	0.183	0.27 ms
100	20	68.5%	0.175	0.27 ms
150	30	69.0%	0.173	0.26 ms
200	40	69.0%	0.171	0.26 ms
300	60	69.3%	0.172	0.23 ms
400	80	69.3%	0.172	0.22 ms

Table 5
Resource comparison. Energy estimates assume 15 W CPU TDP. SLM accuracy values are estimates from published benchmarks. See text for caveats.

Model	**Metric**	**p95 Lat.**	**RAM**	**Energy/q**
Ours	69.0% C	0.18 ms	<10 MB	∼0.0015 mJ
TinyLlama	∼55% F1	∼8 s	∼700 MB	∼120 mJ
Phi-2	∼62% F1	∼15 s	∼1.8 GB	∼225 mJ
Mistral-7B	∼70% F1	∼45 s	∼4.5 GB	∼675 mJ

carefully.

4.5. Comparison with Small Language Models

Table 5 compares our system against quantized SLMs. We note important caveats. SLM accuracy figures are estimated from published benchmarks on SQuAD-style tasks, not directly measured on our benchmark. Latency and memory are estimated from llama.cpp documentation for CPU inference on a standard laptop. Despite these caveats, the order-of-magnitude differences in latency, memory, and energy are robust.

Our system reports answer containment (C) while SLM benchmarks report token F1. These are not directly comparable. Our token F1 (0.171) is lower than all SLMs, which is expected for a sentence-level extractive system. The claim here is not that our system is more accurate. It is that the resource gap (four orders of magnitude in energy, five orders in latency) is large enough that for a meaningful fraction of queries, the extractive system provides a correct answer at negligible cost.

Energy estimation. At 0.097 ms mean latency on a 15 W TDP laptop CPU: $E = 15 \times 0.000097 = 0.0015$ mJ per query. For SLMs, we estimate $P_{\text{CPU}} \times t_{\text{inference}}$: TinyLlama at 8 s on 15 W yields ∼120 mJ. This is simplified and excludes memory subsystem energy. Empirical validation using Intel RAPL is planned as future work.

5. Discussion

5.1. Sustainability Implications

At 0.0015 mJ per query, a classroom of 30 students each asking 100 questions daily would consume approximately 4.5 mJ total. The same workload on Mistral-7B would require ∼2,025 J. Over a school year (200 days), this difference amounts to roughly 405 kJ, or about 0.11 kWh. This is modest in absolute terms but becomes significant across thousands of deployment sites, aligning with the Green AI agenda [12].

5.2. Personalisation Potential

The current evaluation uses a fixed benchmark corpus. However, the modular architecture supports several personalisation extensions relevant to the UMAP community:

User-specific document collections. The chunker (C1) accepts any plain-text documents. In a personalised deployment, each user maintains their own corpus. For example, a student adding lecture

notes throughout a semester, or a healthcare trainee loading clinical guidelines. The system indexes each collection independently, so answers are grounded in documents the user has chosen.

Adaptive confidence thresholds. The refusal threshold θ controls the trade-off between coverage and precision (Table 3). In a personalised system, θ can be adapted per user: a novice might prefer a lower threshold, while an expert might prefer higher.

Cascaded personalisation. In a cascaded design, the extractive system handles high-confidence queries at negligible cost, while low-confidence queries are routed to an LLM for generative personalisation. This is relevant for educational settings explored in LLM4Good work [4, 5], where a study assistant could handle routine lookups extractively and invoke an LLM only for synthesis or explanation.

These extensions are architectural affordances, not experimentally validated features. Implementing and evaluating them is a priority for future work.

5.3. Trustworthiness and Auditability

Every answer is traceable to specific document chunks with character-level offsets. The deterministic pipeline, with fixed seeds, content-addressed chunk IDs, and structured audit logs, ensures reproducibility. This is valuable in educational and healthcare contexts where answer provenance matters [6].

5.4. Limitations

The extractive approach cannot synthesise across passages, rephrase for clarity, or handle questions requiring multi-hop reasoning. These are the cases where LLM-based systems provide genuine value, which is why we frame the system as a first-pass component rather than a replacement.

The 69.0% answer containment means the system misses roughly 3 in 10 answerable questions. Some require inference beyond lexical overlap. Others are cases where the correct chunk is retrieved but the answer sentence is not the top-scoring one. There is room for improvement: the weight ablation (Table 2) shows that even simple changes move containment from 60.8% to 75.0%.

The SLM comparison uses estimated rather than directly measured baselines. We have been transparent about these caveats. The core claim (four orders of magnitude in energy) is robust.

6. Conclusion and Future Work

We presented a sustainable extractive QA system and evaluated it on the full SQuAD v2.0 development set (11,873 questions). The system achieves 69.0% answer containment at a mean latency of 0.097 ms with estimated energy consumption four orders of magnitude lower than quantized SLMs. Ablation studies show the pipeline is robust to chunk size variation and reveal that coverage-only scoring outperforms the multi-factor default at scale. This is an insight not visible in smaller evaluations.

Future work includes: (1) implementing the personalisation extensions in Section 5.2, including a user study with students maintaining their own document collections; (2) integrating SLMs via llama.cpp in a confidence-gated cascade with empirical energy measurements; (3) extending evaluation to additional datasets; (4) hardware-in-the-loop testing on Raspberry Pi and Jetson Nano; and (5) a pilot deployment in a classroom to evaluate personalisation effectiveness in resource-constrained environments.

Acknowledgements

This work was conducted as part of an MSc Individual Project at King's College London, within a larger modular document appliance project. The author thanks the project supervisors for their guidance and support.

Declaration on Generative AI

During the preparation of this work, the author used Claude (Anthropic) in order to: assist with paper drafting and structuring. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the publication's content.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, volume 30, 2017.

[2] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al., LLaMA: Open and efficient foundation language models, arXiv preprint arXiv:2302.13971 (2023).

[3] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in: Advances in Neural Information Processing Systems, volume 33, 2020, pp. 9459–9474.

[4] F. Ghoochani, J. Scharfenberger, B. Funk, R. Doublan, M. J. Odedra, B. Etsiwah, From feedback to formative guidance: Leveraging LLMs for personalized support in programming projects, in: Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '25), ACM, 2025.

[5] A. Piazza, S. Schacht, M. Herzog, Which vocational training program is best for me? Design of a recommender system for school students using large language models, in: Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '25), ACM, 2025.

[6] T. Bano, S. Gupta, Y.-C. Chen, E. Zendejas, S. Krafka, C.-H. Tsai, Exploring responsible use of generative AI in disaster resilience for indigenous communities, in: Adjunct Proceedings of the 33rd ACM Conference on User Modeling, Adaptation and Personalization (UMAP Adjunct '25), ACM, 2025.

[7] Z. Zhang, R. A. Rossi, B. Kveton, Y. Shao, D. Yang, H. Zamani, F. Dernoncourt, J. Barrow, T. Yu, S. Kim, et al., Personalization of large language models: A survey, arXiv preprint arXiv:2411.00027 (2024).

[8] A. Salemi, S. Mysore, M. Bendersky, H. Zamani, LaMP: When large language models meet personalization, in: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 2024, pp. 7370–7392.

[9] P. Zhang, G. Zeng, T. Wang, W. Lu, TinyLlama: An open-source small language model, arXiv preprint arXiv:2401.02385 (2024).

[10] A. Faiz, S. Kaneda, R. Wang, R. Osi, P. Sharma, F. Chen, L. Jiang, LLMCarbon: Modeling the end-to-end carbon footprint of large language models, arXiv preprint arXiv:2309.14393 (2023).

[11] E. Strubell, A. Ganesh, A. McCallum, Energy and policy considerations for deep learning in NLP, arXiv preprint arXiv:1906.02243 (2019).

[12] R. Schwartz, J. Dodge, N. A. Smith, O. Etzioni, Green AI, Communications of the ACM 63 (2020) 54–63.

[13] D. Chen, A. Fisch, J. Weston, A. Bordes, Reading Wikipedia to answer open-domain questions, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, 2017, pp. 1870–1879.

[14] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, W.-t. Yih, Dense passage retrieval for open-domain question answering, arXiv preprint arXiv:2004.04906 (2020).

[15] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter, in: NeurIPS EMC2 Workshop, 2019.

[16] T. Dettmers, M. Lewis, Y. Belkada, L. Zettlemoyer, GPT3.int8(): 8-bit matrix multiplication for transformers at scale, in: Advances in Neural Information Processing Systems, volume 35, 2022.
[17] H. Xu, Y. Li, Y. Luo, H. Li, Z. Zhang, A survey on efficient inference for large language models, arXiv preprint arXiv:2404.14294 (2024).
[18] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al., Holistic evaluation of language models, Annals of the New York Academy of Sciences 1525 (2024) 140–146.
[19] P. Rajpurkar, R. Jia, P. Liang, Know what you don't know: Unanswerable questions for SQuAD, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 2018, pp. 784–789.