AcKnowledge: Acquired Knowledge Representation by Small Language Model Without Pre-training

Sourav Das, Sanjay Chatterji, and **Imon Mukherjee**
Department of Computer Science and Engineering
Indian Institution of Information Technology Kalyani
Kalyani, West Bengal, India
{sourav_phd21, sanjayc, imon}@iiitkalyani.ac.in

Abstract

Large language models (LLMs) are pre-trained on enormous amounts of text data and show acclaimed success in knowledge representation. However, there are two bottlenecks with this approach. (1) Pre-training data cannot be regularly updated once the models are deployed, and it is not very fruitful if the model cannot represent updated knowledge. (2) The consistently increasing size and computational resources make it difficult for non-commercial and individual researchers to fine-tune and scale these language models. Major LLMs with external knowledge are also proprietary. In this paper, we propose AcKnowledge, a framework wrapped around a small, non-pre-trained language model for an open-domain question-answering (QA) experiment. AcKnowledge learns relevant knowledge from the internet via meta-learning based on user questions, and re-learns from user feedback if knowledge is misrepresented. Our efficient knowledge representation framework avoids pre-training overhead while enabling updated information. Benchmarking shows competitive performance against similarly sized state-of-the-art (SoTA) LLMs on gold standard QA datasets, demonstrating the potential of integrating internet search and user feedback for improved performance and generalizability. The repository of the work is available at https://github.com/SouravD-Me/AcKnowledge---KnowledgeLM-ACL-2024.

1 Introduction

The excellent performance of large language models (LLMs) in various natural language processing (NLP) tasks can be mainly attributed to their ability to capture and represent knowledge from extensive pre-training on massive text corpora (Chang et al., 2024; Min et al., 2023). However, the outdated nature of data for pre-trained knowledge can limit their adaptability to new information or recent events (Kazemnejad et al., 2023). Traditional methods for updating LLMs, such as continuous pre-training on the latest data or fine-tuning, are computationally expensive and time-consuming (Tian et al., 2023).

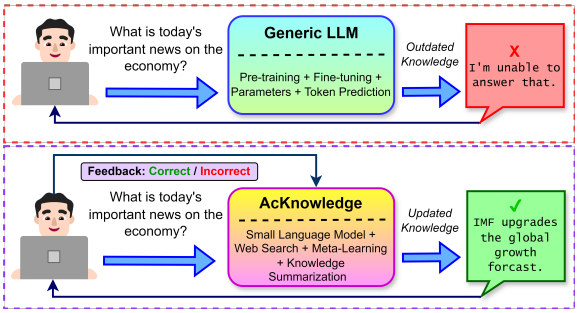

Figure 1: Fundamental illustration of AcKnowledge: Representing acquired knowledge through user questions and feedback.

To address these limitations, we propose AcKnowledge, a novel framework that equips a small language model (SLM) with the dynamic capability to acquire and represent knowledge without conventional pre-training. Our approach exploits real-time web search and meta-learning (Xie et al., 2023; Li et al., 2020) to enable an SLM to learn new information efficiently. Upon receiving a user's question, the topics are extracted using Latent Dirichlet Allocation (LDA) (Blei et al., 2003). These topics are then transmitted to the language model (LM), which uses these topics as keywords to perform a targeted online search and uses meta-learning (Lin and Chen, 2020) to acquire relevant knowledge. The acquired knowledge is then summarized (Moratanch and Chitrakala, 2017) and presented to the user as a concise answer.

AcKnowledge also integrates a user feedback mechanism to ensure authenticity and reliability for generated answers. Negative feedback triggers a new search iteration to find more accurate information, while positive feedback fortifies the learning.

Proceedings of the 1st Workshop on Towards Knowledgeable Language Models (KnowLLM 2024), pages 83–95

User feedback plays a vital role in AcKnowledge's learning loop in augmenting the acquired knowledge for increasingly more correct and factual answers. We evaluate AcKnowledge's performance by benchmarking it against similarly sized LMs in open-domain QA tasks, demonstrating competitive results despite the absence of traditional pre-training.

The main contributions of this paper are:

- We propose AcKnowledge, a novel open-domain QA system that utilizes a non-pre-trained SLM to dynamically acquire and represent knowledge from the internet based on user questions.
- A meta-learning algorithm is implemented to enable the language model to efficiently learn from search results and refine its knowledge representation through user feedback.
- The framework is designed for users to initiate re-searching for answers if the initial response is misrepresented, enhancing the reliability and user control over language model outputs.
- The effectiveness of AcKnowledge is demonstrated through extensive benchmarking against similar language models, showcasing competitive performance without relying on pre-training.
- The quality of the generated answers is meticulously analyzed, showcasing the impact of real-time knowledge acquisition in adaptable SLMs for efficient QA.

2 Relevant Works

Knowledge representation is indispensable for NLP systems to understand meaning and perform reasoning. The statistical approaches in the early last decade like word embeddings (Mikolov et al., 2013; Chen et al., 2013) learned vector representations but lacked explicit knowledge modeling. Further advances in integrated neural networks with symbolic knowledge graphs and ontologies through techniques like graph convolutional networks (Kipf and Welling, 2016).

Hybrid neuro-symbolic methods show promise in injecting knowledge into large pre-trained language models like RoBERTa (Liu et al., 2019) to improve common sense reasoning (Bosselut et al., 2019) and factual grounding (Guan et al., 2020). Multimodal learning from transformer architecture has also been in research focus (Tan and Bansal, 2019). Key challenges in representing knowledge often include effective representation and context-sensitivity to the core topic (Verma and Bergler, 2023), performing reasoning over learned representations (Saha et al., 2022), and generating logical forms (Hu et al., 2022). Promising directions also involve meta-learning for fast knowledge adaptation (Zhao et al., 2022) and graph embedding methods for knowledge representation (Cao et al., 2024).

3 System Framework

Our fundamental objective is to develop AcKnowledge with the ability to dynamically retrieve and adapt relevant knowledge seamlessly from the internet. The proposed system comprises several key components that work in tandem to facilitate this process. The overview of our system is illustrated in Figure 2. The primary components of the framework are discussed in the following sections.

3.1 Answer Retrieval from Internet Search

To accumulate external knowledge from the internet, our approach employs a two-stage information retrieval process. First, LDA is implemented for topic extraction from the user question. LDA serves as an unsupervised clustering model for the revelation of topics in a collection of documents (Alhawarat and Hegazi, 2018; Zong et al., 2021). It can be formalized as a probabilistic generative model. In this model, the distribution of topics for any number of questions can be represented as $\Omega_Q \approx \text{Dirichlet}(\delta)$, where Ω is the distribution parameter. The Dirichlet distribution is used here to guide the distribution of topics from tokens, and the parameter δ controls the sparsity of the distribution. Second, these topic words are transmitted to the language model. Using these topics as keywords, it uses the Google search API to retrieve a set of relevant passages, such as $\mathbf{P}$ from the search results.

Here, we employ a dense passage retrieval technique (Karpukhin et al., 2020) to rank and select the most relevant passages. We encode each passage $\mathbf{P}_i$ to obtain a sequence of dense vector representations $\mathbf{P}_i = (\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n)$, where $\mathbf{p}_n \in \mathbb{R}^{d_{\text{emb}}}$. Here, $\mathbb{R}^{d_{\text{emb}}}$ is used for the dimensional embedding in real space of the *n-th* token in $\mathbf{P}_i$. Hereafter, these passages are passed through the meta-learning module to learn from them as potential answers to the users' questions.

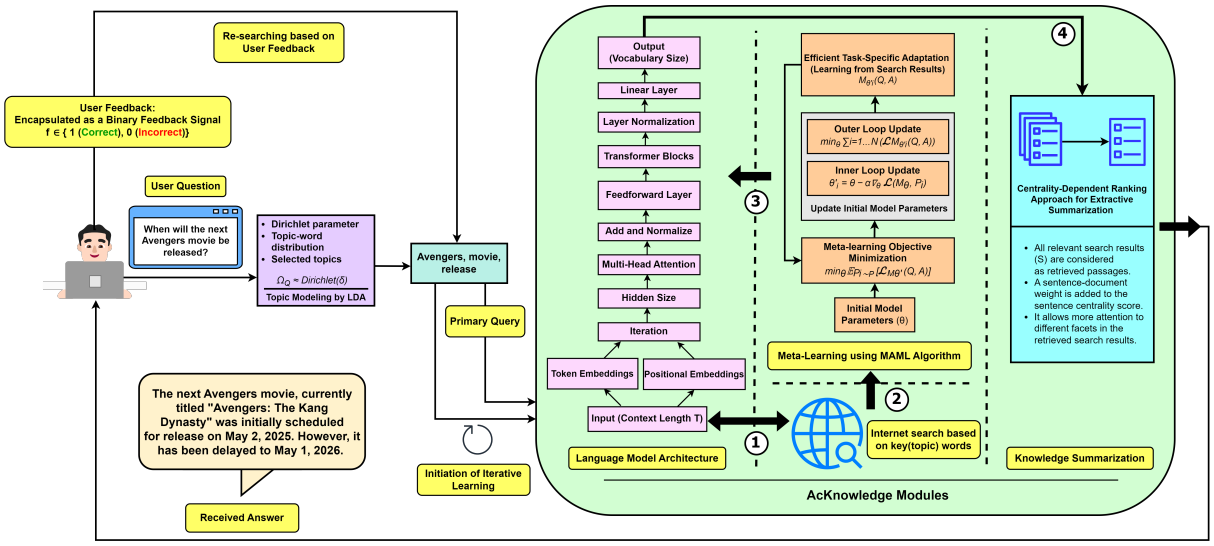

Figure 2: Overview of AcKnowledge. The user asks a question; keywords are extracted and used by the language model to search the internet. The MAML algorithm learns passages from the search results to retrieve potential answers. These passages are then transferred to the language model. It generates an answer summary, on which extractive summarization is performed to finally present a concise answer. If incorrect, the user can provide feedback to initiate iterative learning for improved responses.

3.2 Meta-learning for Search Results

To effectively utilize the information retrieved, we employ a meta-learning algorithm that learns from the retrieved passages. We use the model-agnostic meta-learning algorithm (MAML) (Lee et al., 2022), which has shown promising results in natural language understanding (NLU) scenarios.

For retrieving information from the passages $\mathbf{P}$, the proposed language model $\mathcal{M}$ adapts to these passages from meta-learning. As $\mathbf{P}_i$ represents a retrieved passage, the aim is to accumulate the sequence of texts from it and send it to $\mathcal{M}$. Adapt all passages from the search results for each question, by minimizing the meta-objective $\min_\theta \mathbb{E}_{\mathbf{P}_i \sim \mathbf{P}} \left[\mathcal{L}_{\mathcal{M}_{\theta'}}(Q, A) \right]$. Here, θ represents the parameters of the language model, $\mathbb{E}$ is the expected value of the loss function for the distribution of passages from search results, and $\mathcal{L}$ denotes the cross-entropy loss function. The meta-learning algorithm updates the model parameters θ by taking a gradient step on each passage $\mathbf{P}_i$, resulting in adapted parameters θ'. The adapted model $\mathcal{M}_{\theta'}$ is then evaluated on the original question-answer pair (Q, A). Internally, the meta-learning process is further decomposed into two stages; the inner loop and the outer loop. In the inner loop, for each search result in $\mathbf{P}_i$, the model parameters θ are updated using gradient descent to minimize the loss specific to the task $\mathcal{LM}\theta(\mathbf{P}_i)$:

$$\theta' i = \theta - \alpha \nabla_\theta \mathcal{L}(\mathcal{M}_\theta, \mathbf{P}_i) \quad (1)$$

This inner loop uses the learning rate α to update itself and allows the model to adapt to the information contained in the passages. The updated parameters θ'_i are specific to each passage $\mathbf{P}_i$.

In the outer loop, the meta-objective is optimized across all search results:

$$\min_\theta \sum_{i=1}^{N} \mathcal{LM}_{\theta'_i}(Q, A) \quad (2)$$

The outer loop update aggregates the losses from the adapted models $\mathcal{M}_{\theta'_i}$ and updates the global parameters θ to minimize the expected loss across all search results.

By iterating between the updates of the inner loop and the outer loop, meta-learning enables the language model to efficiently incorporate the retrieved passages from the search results and generalize to unseen questions. The adapted parameters θ'_i capture the question-specific information from each search result, while the global parameters θ learn to adapt to new information.

The MAML algorithm emphasizes learning a good initialization of the model parameters that can rapidly adapt to new questions with just a few gradient steps. This is particularly advantageous in QA tasks, where the framework must quickly

assimilate relevant information from the search results to generate accurate answers without relying on extensive pre-training.

3.3 Language Model Development

We build the language model in the AcKnowledge framework from scratch based on the fundamental transformer architecture (Vaswani et al., 2017). The model consists of a parameter size of just 125 Million, with a multi-layer encoder and decoder, with each layer employing multi-head self-attention mechanisms to capture long-range dependencies in text sequences.

The model encoder processes the input keywords from a question $\mathcal{K} = \{k_1, k_2, \ldots, k_n\}$, while the decoder generates the corresponding answer $A = \{a_1, a_2, \ldots, a_m\}$. This process is enhanced by meta-learning from retrieved passages. Both the encoder and decoder consist of multiple layers, each containing a multi-head self-attention sublayer and a position-wise feedforward sublayer.

In this architecture, the input sequence $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_T)$ is mapped to token embeddings, with T representing the length of the sequence. The encoder transforms this input into a sequence of continuous representations $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_i)$, which the decoder then uses to generate the output sequence $Y = (y_1, y_2, \ldots, y_m)$.

The input token embeddings are the sum of token embeddings and positional embeddings:

$$\boldsymbol{x}_T = \boldsymbol{W}_{tok}[x_T] + \boldsymbol{W}_{pos}[T], \tag{3}$$

where $\boldsymbol{W}_{tok}$ and $\boldsymbol{W}_{pos}$ are embedding matrices for tokens and positions, respectively.

Multi-head attention is a key component, calculated in multiple heads H. Each head computes attention scores using query, key, and value projections:

$$\boldsymbol{A}^{(h)} = \text{Softmax}\left(\frac{\boldsymbol{Q}^{(h)}{\boldsymbol{K}^{(h)}}^{\top}}{\sqrt{d_k}}\right)\boldsymbol{V}^{(h)}, \tag{4}$$

where $\boldsymbol{Q}^{(h)}, \boldsymbol{K}^{(h)}, \boldsymbol{V}^{(h)}$ are the projections, and d_k is the dimension of each head.

These attention scores are concatenated and linearly projected:

$$\text{MultiHead}(\boldsymbol{x}) = \text{Cat}(\boldsymbol{A}^{(1)}, \boldsymbol{A}^{(2)}, \ldots, \boldsymbol{A}^{(H)})\boldsymbol{W}_O, \tag{5}$$

where $\boldsymbol{W}_O$ is the projection matrix.

The position-wise feed-forward network (FFN) processes each token independently:

$$\text{FFN}(\boldsymbol{x}) = \max(0, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{6}$$

with learnable parameters $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2$.

Both the attention and feed-forward layers use residual connections and layer normalization:

$$\boldsymbol{x}' = \text{LayerNorm}(\boldsymbol{x} + \text{MultiHead}(\boldsymbol{x})), \tag{7}$$

$$\boldsymbol{x}'' = \text{LayerNorm}(\boldsymbol{x}' + \text{FFN}(\boldsymbol{x}')), \tag{8}$$

ensuring stability and efficiency.

The final output, after layer normalization and a linear projection, provides the logits for next-token prediction:

$$\boldsymbol{y} = \text{LayerNorm}(\boldsymbol{x}'')\boldsymbol{W}_y + \boldsymbol{b}_y, \tag{9}$$

with $\boldsymbol{W}_y$ and $\boldsymbol{b}_y$ as learnable parameters.

The model minimizes the cross-entropy loss between predicted logits $\boldsymbol{y}$ and target tokens $\boldsymbol{t}$:

$$\mathcal{L} = -\sum_{i=1}^{T} \log p(t_i|\boldsymbol{x}, \boldsymbol{\theta'}), \tag{10}$$

where $\boldsymbol{\theta'}$ represents all learnable parameters of the model from meta-learning, and $p(t_i|\boldsymbol{x}, \boldsymbol{\theta'})$ is the predicted probability of the *i-th* target token given the input sequence $\boldsymbol{x}$ and the model parameters $\boldsymbol{\theta'}$.

3.4 Knowledge Summarization

After incorporating the retrieved information through meta-learning, the language model generates a summary or answer to the user's query. To ensure that the generated answer is concise, we employ extractive summarization. Here, the most salient sentences from the summary are selected and combined to form the final answer. We use the centrality-dependent ranking approach (Liang et al., 2021) to identify and rank the most important text segments.

Here, a sentence-document weight is added to the sentence centrality score, allowing more attention to different facets in the generated answer summaries. This approach is applicable for long and multi-document scenarios, as in our experiment the summaries with multiple information generated by the language model. These text segments are then merged and represented as the final answers. We do not paraphrase the answers by employing abstractive summarization. This reduces the scope for representing misinformation (Krishna et al., 2024). Finally, extractive summarized content is presented to users as final answers. Table 1 represents question-answer pairs in 3 distinct aspects.

QA 1: Topic → Latest News and Events
Q: What is the latest news development for today?
(**Question without any specific date or timeline guidance.**)
A: As per the current time of Thursday, May 23, 2024, 09:34 UTC, a devastating tornado outbreak struck Iowa, leaving a trail of destruction. At least five people were confirmed dead and more than 35 injured. The town of Greenfield, southwest of Des Moines, bore the brunt of the damage, with homes destroyed, trees uprooted, and even wind turbines twisted by the powerful winds. Relief efforts are underway to assist those affected by the disaster.
Majority Feedback (5:0): Correct. (Accurate date and news representation.)
QA 2: Topic → Education and Training
Q: Suggest some NLP courses offered by leading universities.
(**No mention of ranking, regionality, or instructor names which can initiate potential bias.**)
A: Some of the most recognized NLP courses offered by leading universities are Natural Language Processing with Deep Learning: Stanford University CS224n, Computational Linguistics: University of California, Berkeley, Natural Language Understanding with Deep Learning: Massachusetts Institute of Technology (MIT), Natural Language Processing: University of Michigan.
Majority Feedback (3:2): Correct. (Factually correct, course choices can be subjective for different people.)
QA 3: Topic → Science and Research
Q: What is the fundamental advantage of stochastic gradient descent over batch gradient descent?
(**Extensive machine learning insight is required to answer this question.**)
A: The fundamental advantage of stochastic gradient descent (SGD) compared to batch gradient descent is its ability to converge more quickly and with higher computational efficiency, particularly for large datasets. This efficiency occurs from SGD updating the model parameters after examining examples or mini-batch, while batch gradient descent waits to update the parameters until the complete training dataset is processed.
Majority Feedback (4:1): Partially Correct. (The presented knowledge in the answer is not completely accurate. Here, higher and examples are not the correctly predicted tokens, rather they should be **better** and **each example**, respectively. However, the rest of the answer is correct.)

Table 1: Question-answer pairs to demonstrate the real-time response of AcKnowledge. Different topics of questions elaborate on AcKnowledge's prowess in efficiently acquiring and presenting updated knowledge in answers.

3.5 Iterative Learning from User Feedback

Our approach implements user feedback in an iterative learning process to improve the quality of the generated answers and adapt to user preferences. After receiving an answer from the language model, the user can provide positive or negative feedback, indicating their satisfaction with the response.

If the users provide negative feedback, the system initiates another internet search, using the same keywords from the original query. The retrieved information and the language model's parameters are updated based on the user's feedback to generate a more accurate and relevant answer.

This iterative process continues until users are satisfied with the authenticity of the information in the answer. User feedback is a valuable indication that guides AcKnowledge in enhancing its understanding and generating more accurate responses with correct information.

For the user feedback mechanism, a group study is carried out. Here, the answers are evaluated by a group of 5 people consisting of 2 NLP experts, 2 researchers, and a student. These people are only the end users and are not involved in any of the experiments described in the paper.

After receiving an answer, the users individually provide binary feedback signals $f \in \{1, 0\}$, indicating satisfaction with the answer. If any feedback is labeled as $(f = 0)$, i.e., 'incorrect', $\mathcal{M}$ initiates a new search process, searching for the unexplored search results in the previous hop, and updates its parameters using the MAML algorithm described earlier. After any *n-th* iteration of iterative learning and refinement, the users provide positive feedback, indicating that the generated answer is correct and high quality. The majority of user group feedback determines each answer's correctness or incorrectness. The user feedback on 3 distinct aspects is shown in Table 1.

4 Experiments

To evaluate AcKnowledge's performance on gold-standard corpora, we use two widely-used open-domain QA datasets; Stanford Question Answering Dataset (SQuAD 2.0) (Rajpurkar et al., 2018) and Natural Questions (NQ) (Kwiatkowski et al., 2019), SQuAD 2.0 integrates approximately 50,000 adversarial-designed unanswerable questions to mimic responsive questions. For good performance on SQuAD 2.0, ideally, systems should recognize when the text does not support a response and refrain from responding. NQ is a large-scale dataset with more than 300,000 question-answer pairs based on real-world Google search queries.

It includes a diverse range of topics and question types, with an average of 16.5 tokens per question and 77 tokens per answer.

We preprocess the datasets by tokenizing the text using the WordPiece tokenizer (Hussain et al., 2023) and converting the tokens to their corresponding embeddings using Word2vec (Mikolov et al., 2013). The preprocessed datasets are then split into training, validation, and test sets with a ratio of 80%, 10%, and 10%, respectively.

4.1 Experimental Settings

We implement AcKnowledge using the PyTorch framework (Paszke et al., 2019). The encoder and decoder of $\mathcal{M}$ consist of 6 layers each, with a hidden size of 768 and 8 attention heads. The model is trained in gold standard QA corpora using the Adam optimizer (Kingma, 2014) with a learning rate of 0.0005 and a batch size of 32. The maximum sequence length is set to 1024 tokens. We apply gradient clipping with a maximum norm of 1.0 to stabilize the training.

For the internet search component, we use the Google search API to retrieve the first 10 search results sequentially for any question. The MAML algorithm is applied with a learning rate $\alpha = 0.001$ and a maximum of 5 adaptation steps. If a user initiates negative feedback for any answer, this process is repeated for iterative learning.

4.2 Quantitative Evaluation Metrics

We evaluate the performance of our approach using standard empirical evaluation metrics. For such purpose, we employ metrics such as semantic fluency (*Sem-FL*) (Zhou et al., 2023) for quantifying the semantic coherence and meaningfulness of the outputs, Length fluency (*Len-FL*) (Zhou et al., 2023) for evaluating the ability to generate outputs of appropriate verbosity, *F1* score (Tan et al., 2016), computed by comparing the word-level overlap between the predicted and ground truth answers, binary accuracy (*Y/N*) (Choi et al., 2018), for verifying the accuracy in binary answerable questions, exact match (*EM*) (Chen et al., 2024), the percentage of predictions that exactly match the ground truth answer, BLEU (*BL*) (Chen et al., 2023), the metric widely used in machine translation that measures the n-gram overlap between the predicted and reference answers, ROUGE (*RG*) (Schluter, 2017), the metric commonly used for summarization tasks, which evaluates the quality of the generated summaries based on n-gram overlap with reference summaries, and METEOR (*MR*) (Chen et al., 2019), for analyzing multiple matching criteria, including exact word matches, stemmed word matches, synonyms, and paraphrases.

Furthermore, we perform statistical significance tests such as the Wilcoxon signed-rank test (Narayan et al., 2023) to determine if the performance differences between the SoTA models and AcKnowledge are statistically significant (ref. Figure 3**(b)**). This significance is measured using the p value. Let $\mu_{\mathcal{LM}}$ and $\mu_{\mathcal{M}}$ denote the mean scores for the SoTA models and AcKnowledge, respectively. The null hypothesis H_0 for the performance below the significance threshold and the alternative hypothesis H_1 for the performance above the significance threshold are defined as follows:

$$H_0 : \mu_{\mathcal{LM}} = \mu_{\mathcal{M}} \tag{11}$$

$$H_1 : \mu_{\mathcal{LM}} > \mu_{\mathcal{M}} \tag{12}$$

We set a standard significance threshold for p value (0.05) (Smucker et al., 2007) and calculate to determine below and above-significance threshold performance for all the SoTA models compared with AcKnowledge.

4.3 SoTA and Baseline Comparisons

The performance of AcKnowledge is compared with several state-of-the-art (SoTA) models for QA. Small language models are scarce for downstream tasks. However, several SoTA models are selected based on comparable parameter sizes and considering their efficiency in QA. Such models are BLOOM (Muennighoff et al., 2022), Open Pretrained Transformer (OPT) by Meta AI (Zhang et al., 2022), ELECTRA(Clark et al., 2019), Fine-tuned LAnguage Net (Flan-T5-base) by Google (Chung et al., 2024), DeBERTaV3-Base (He et al., 2022), GPT-Neo (Kashyap et al., 2022), and MiniLM (Wang et al., 2020).

We do not explicitly fine-tune these models, but rather deploy with their recommended setup with default hyperparameters. To benchmark the performance of SoTA models and AcKnowledge, we compare this evaluation setup against several strong baseline models, such as BERT (Devlin et al., 2018) for checking the performance based on pre-training and fine-tuning on the QA datasets without any external knowledge or user feedback, SpanBERT (Joshi et al., 2020) for evaluating the span of answers against each question, and RAG-Base (Braunschweiler et al., 2023) for comparing

Models	Sem-FL	Len-FL	F1	Y/N-Ac	Ex-M	BL	RG	MR
Dataset: SQuAD 2.0								
BLOOM 350M	0.84	0.91	88.0	82.1	70.9	87.4	87.9	86.7
OPT 350M	0.85	0.92	87.7	81.6	70.5	87.0	87.6	86.3
ELECTRA 335M	0.82	0.89	87.3	81.0	69.9	86.7	87.2	85.9
Flan T5-Base 250M	0.83	0.90	86.9	80.4	69.3	86.3	86.8	85.5
DBV3-Base *w/* Voc. 134M	0.80	0.87	86.5	79.8	68.7	85.9	86.4	85.1
GPT-Neo 125M	0.81	0.88	86.1	79.2	68.1	85.5	86.0	84.7
MiniLM-XLMR 117M	0.79	0.86	85.7	78.6	67.5	85.1	85.6	84.3
AcKnowledge 125M	**0.88**	**0.95**	**89.5**	**84.2**	**73.1**	**88.9**	**89.4**	**88.2**
BERT-base	0.78	0.85	85.5	78.2	67.1	84.9	85.4	84.1
SpanBERT	0.77	0.84	85.1	77.6	66.5	84.5	85.0	83.7
RAGBase	0.76	0.83	84.7	77.0	65.9	84.1	84.6	83.3
Dataset: NQ								
BLOOM 350M	0.83	0.90	87.6	81.7	70.3	86.9	87.5	86.2
OPT 350M	0.84	0.91	87.2	81.1	69.7	86.5	87.1	85.8
ELECTRA 335M	0.81	0.88	86.8	80.5	69.1	86.1	86.7	85.4
Flan T5-Base 250M	0.82	0.89	86.4	79.9	68.5	85.7	86.3	85.0
DBV3-Base *w/* Voc. 134M	0.79	0.86	86.0	79.3	67.9	85.3	85.9	84.6
GPT-Neo 125M	0.80	0.87	85.6	78.7	67.3	84.9	85.5	84.2
MiniLM-XLMR 117M	0.78	0.85	85.2	78.1	66.7	84.5	85.1	83.8
AcKnowledge 125M	**0.87**	**0.94**	**89.0**	**83.7**	**72.3**	**88.4**	**88.9**	**87.7**
BERT-base	0.77	0.84	85.0	77.7	66.3	84.3	84.9	83.6
SpanBERT	0.76	0.83	84.6	77.1	65.7	83.9	84.5	83.2
RAGBase	0.75	0.82	84.2	76.5	65.1	83.5	84.1	82.8

Table 2: Benchmark evaluation on SQuAD 2.0 and NQ datasets. The original parameter size of the DeBERTaV3 model is 86 Million. However, for all DeBERTaV3 models, the vocabulary size is 128K tokens, adding approximately 48 Million parameters to the total size. *M* is used to denote the parameter size of each model in Million.

the performance of vanilla models (including ours) with retrieval augmented generation (RAG)-based language model.

5 Results and Discussion

5.1 Benchmarking Results

In this section, we present the results of our proposed approach and compare it with various baselines and SoTA models. We also provide a qualitative analysis of the generated answers, the impact of user feedback on model performance, and the limitations and potential improvements of our approach.

Table 2 presents a comprehensive evaluation of SoTA and baseline language models on SQuAD 2.0 and NQ. The results demonstrate that AcKnowledge consistently outperforms the other models across both datasets and all evaluation metrics. Despite having a smaller parameter size compared to the other models, AcKnowledge achieves superior performance, highlighting its efficiency and effectiveness in QA tasks. The bold values in the table emphasize the superior performance of AcKnowledge.

Among the baseline models, the BERT-base exhibits the best performance, followed by SpanBERT and RAGBase. However, their performance falls short of that of the SoTA models, indicating the more recent advances made in QA using such models.

5.2 Qualitative Analysis

Scrutinizing further into the SoTA comparisons, a human evaluation is carried to assess the quality of the generated answers using several qualitative metrics. A blind evaluation of the answers of all models on gold standard datasets is performed by the same group of 5 people described earlier. Here, 100 questions are randomly selected from the test sets, and the quality of the answers generated by each model is manually evaluated.

The qualitative evaluation metrics are the maximum token length ($Max\text{-}T_k$) supported by each model, context preservation in answers ($C\text{-}P_r$), correctness (C_r) (Falke et al., 2019), and completeness (C_n) (Lu et al., 2022). Context preservation is a binary metric indicating whether the model can maintain the context of the question when generating answers. Correctness measures the accuracy of the generated answers, while completeness evaluates the extent to which the answers cover all the necessary information.

The results in Table 3 demonstrate that AcKnowledge performs best in correctness and completeness. The table also highlights the importance of context preservation in QA tasks for bet-

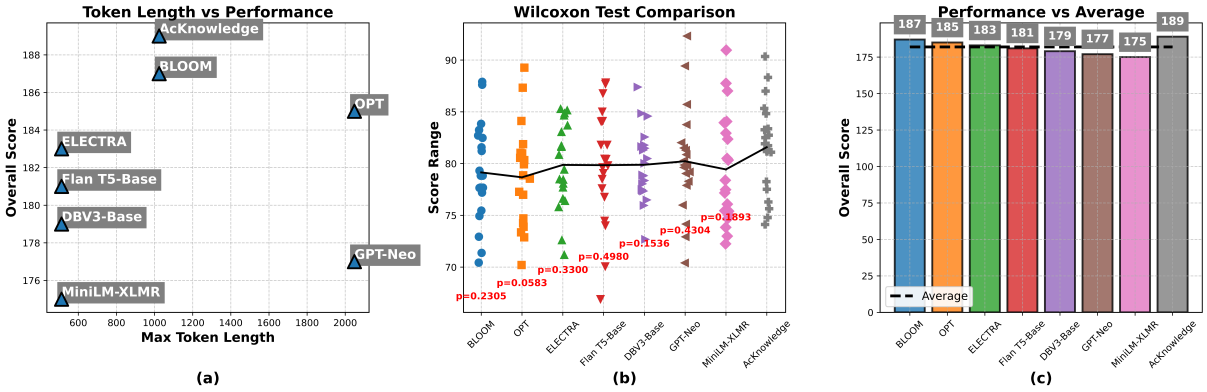

Figure 3: Left **(a)**: Balance between token length and answering performance of models. Middle **(b)**: Wilcoxon test scores compared to AcKnowledge, showing all models perform above the significance level. The black line connects mean performance scores, indicating AcKnowledge and GPT-Neo perform similarly and better than others. Right **(c)**: Cumulative C_r and C_n performance scores of the models against their combined average performance threshold.

ter answer responses. Apart from ELECTRA and MiniLM-XLMR, the rest of the models generally perform better in correctness and completeness. This depicts that maintaining context from the users' questions plays a crucial role in high-quality and relevant answers.

Models	Max-T_k	C-P_r	C_r	C_n
BLOOM	1024	✓	0.92	0.95
OPT	2048	✓	0.91	0.94
ELECTRA	512	×	0.90	0.93
Flan T5-Base	512	✓	0.89	0.92
DBV3-Base	512	✓	0.88	0.91
GPT-Neo	2048	✓	0.87	0.90
MiniLM-XLMR	512	×	0.86	0.89
AcKnowledge	**1024**	✓	**0.94**	**0.95**

Table 3: Qualitative analysis of the compared models on SQuAD 2.0 and NQ. We show the standard token lengths that are mentioned in each model documentation.

5.3 Ablation Studies

We conduct ablation studies to assess the impact of meta-learning and user feedback in the AcKnowledge architecture.

Meta-Learning. We replace the MAML algorithm with standard fine-tuning to evaluate its significance. Results show that with meta-learning, AcKnowledge achieves 89.5% F1 and 84.2% exact match. However, without meta-learning, scores drop to 85.8% and 80.1% respectively. This highlights the crucial role of meta-learning in efficiently incorporating retrieved information and adapting to unseen questions.

User Feedback. Disabling the feedback loop and iterative learning process resulted in reduced performance. With user feedback, AcKnowledge achieves 87.2% F1 and 83.7% exact match, whereas, without it, scores decrease to 82.6% and 77.3% respectively. Incorporating user feedback significantly enhances the framework's understanding and answer accuracy.

These findings underscore the importance of integrating meta-learning and user feedback in knowledge representation in QA by a small language model.

6 Conclusion

We introduced AcKnowledge, a framework that can search the internet for updated answers to user questions, learn from the search results using meta-learning, and assimilate user feedback to improve performance. Our proposed approach outperforms various SoTA and baseline models in standard QA evaluation metrics. Our approach has several potentials for language model applications. Firstly, it demonstrates the benefit of integrating internet search and meta-learning in language models to improve their answering ability. Secondly, it can also answer complex questions that require access to up-to-date and diverse information sources. Third, it can be used to develop scalable language models that can learn from user feedback to improve their performance and adapt to user preferences. There are prominent future research directions for our work. We aim to explore the prospect of scalability and robustness in larger and real-world deployable scenarios. This experiment can pave the way for developing more knowledgeable language models that can assist users in various tasks and scenarios.

References

Mohammad Alhawarat and M Hegazi. 2018. Revisiting k-means and topic modeling, a comparison study to cluster arabic documents. *IEEE Access*, 6:42740–42749.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4762–4779.

Norbert Braunschweiler, Rama Doddipatla, Simon Keizer, and Svetlana Stoyanchev. 2023. Evaluating large language models for document-grounded response generation in information-seeking dialogues. In *Proceedings of the 1st Workshop on Taming Large Language Models: Controllability in the era of Interactive Assistants!*, pages 46–55.

Jiahang Cao, Jinyuan Fang, Zaiqiao Meng, and Shangsong Liang. 2024. Knowledge graph embedding: A survey from the perspective of representation spaces. *ACM Computing Surveys*, 56(6):1–42.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2024. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In *Proceedings of the 2nd workshop on machine reading for question answering*, pages 119–124.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17754–17762.

Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2013. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*.

Yirong Chen, Xiaofen Xing, Jingkai Lin, Huimin Zheng, Zhenyu Wang, Qi Liu, and Xiangmin Xu. 2023. Soulchat: Improving llms' empathy, listening, and comfort abilities through fine-tuning with multi-turn empathy conversations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1170–1183.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. Quac: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2019. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Tobias Falke, Leonardo FR Ribeiro, Prasetya Ajie Utama, Ido Dagan, and Iryna Gurevych. 2019. Ranking generated summaries by correctness: An interesting but challenging application for natural language inference. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 2214–2220.

Jian Guan, Yotam Perl, Caglar Gulcehre, Daniel Bieber, Ashish Vaswani, Matthew Johnson, and Charles Sutton. 2020. A unified mrc framework for named entity recognition. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5849–5859.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2022. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*.

Xixin Hu, Xuan Wu, Yiheng Shu, and Yuzhong Qu. 2022. Logical form generation via multi-task learning for complex question answering over knowledge bases. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1687–1696.

Yasir Hussain, Zhiqiu Huang, Yu Zhou, Izhar Ahmed Khan, Nasrullah Khan, and Muhammad Zahid Abbas. 2023. Optimized tokenization process for open-vocabulary code completion: An empirical study. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, pages 398–405.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S Weld, Luke Zettlemoyer, and Omer Levy. 2020. Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the association for computational linguistics*, 8:64–77.

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.

Rohan Kashyap, Vivek Kashyap, et al. 2022. Gpt-neo for commonsense reasoning–a theoretical and practical lens. *arXiv preprint arXiv:2211.15593*.

Amirhossein Kazemnejad, Mehdi Rezagholizadeh, Prasanna Parthasarathi, and Sarath Chandar. 2023. Measuring the knowledge acquisition-utilization gap in pretrained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 4305–4319.

DP Kingma. 2014. Adam: a method for stochastic optimization. In *Int Conf Learn Represent*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2024. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36.

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*, 7:453–466.

Hung-Yi Lee, Shang-Wen Li, and Thang Vu. 2022. Meta learning for natural language processing: A survey. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 666–684.

Jing Li, Billy Chiu, Shanshan Feng, and Hao Wang. 2020. Few-shot named entity recognition via meta-learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(9):4245–4256.

Xinnian Liang, Shuangzhi Wu, Mu Li, and Zhoujun Li. 2021. Improving unsupervised extractive summarization with facet-aware modeling. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1685–1697.

Chong-En Lin and Kuan-Yu Chen. 2020. A preliminary study on using meta-learning technique for information retrieval. In *Proceedings of the 32nd Conference on Computational Linguistics and Speech Processing (ROCLING 2020)*, pages 59–71.

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. *arXiv preprint arXiv:1907.11692*.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Stefano Markidis, Steven Wei Der Chien, Erwin Laure, Ivy Bo Peng, and Jeffrey S Vetter. 2018. Nvidia tensor core programmability, performance & precision. In *2018 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*, pages 522–531. IEEE.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. 2023. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40.

N Moratanch and S Chitrakala. 2017. A survey on extractive text summarization. In *2017 international conference on computer, communication and signal processing (ICCCSP)*, pages 1–6. IEEE.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng-Xin Yong, Hailey Schoelkopf, et al. 2022. Crosslingual generalization through multitask finetuning. *arXiv preprint arXiv:2211.01786*.

Shashi Narayan, Joshua Maynez, Reinald Kim Amplayo, Kuzman Ganchev, Annie Louis, Fantine Huot, Anders Sandholm, Dipanjan Das, and Mirella Lapata. 2023. Conditional generation with a question-answering blueprint. *Transactions of the Association for Computational Linguistics*, 11:974–996.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

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.

Amrita Saha, Shafiq Joty, and Steven CH Hoi. 2022. Weakly supervised neuro-symbolic module networks for numerical reasoning over text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11238–11247.

Natalie Schluter. 2017. The limits of automatic summarisation according to rouge. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 41–45. Association for Computational Linguistics.

Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 623–632.

Hao Tan and Mohit Bansal. 2019. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5100–5111.

Ming Tan, Cicero Dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved representation learning for question answer matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 464–473.

Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. 2023. Fine-tuning language models for factuality. In *The Twelfth International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Harsh Verma and Sabine Bergler. 2023. Clac at semeval-2023 task 2: Comparing span-prediction and sequence-labeling approaches for ner. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 1558–1561.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

Hongtu Xie, Jiaxing Chen, Yiquan Lin, Lin Zhang, Guoqian Wang, and Kai Xie. 2023. External knowledge document retrieval strategy based on intention-guided and meta-learning for task-oriented dialogues. *Advanced Engineering Informatics*, 56:102020.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yi Zhang, Zhongyang Yu, Wanqi Jiang, Yufeng Shen, and Jin Li. 2023. Long-term memory for large language models through topic-based vector database. In *2023 International Conference on Asian Language Processing (IALP)*, pages 258–264. IEEE.

Yiren Zhao, Xitong Gao, Ilia Shumailov, Nicolo Fusi, and Robert Mullins. 2022. Rapid model architecture adaption for meta-learning. *Advances in Neural Information Processing Systems*, 35:18721–18732.

Wangchunshu Zhou, Yuchen Eleanor Jiang, Ethan Wilcox, Ryan Cotterell, and Mrinmaya Sachan. 2023. Controlled text generation with natural language instructions. In *International Conference on Machine Learning*, pages 42602–42613. PMLR.

Chengqing Zong, Rui Xia, and Jiajun Zhang. 2021. Topic model. In *Text Data Mining*, pages 145–162. Springer.

A Appendix

A.1 Ethical Statement

Although our proposed framework achieves promising results, there are currently a few limitations and potential areas for improvement. First, the performance of the framework may be affected by the quality and relevance of the search results. Search results can certainly integrate incorrect or misleading information. User feedback can preliminarily mitigate that by using 'incorrect' feedback. Implementing an ensemble method for more filtered information retrieval techniques could help mitigate this issue even further. Second, the framework's ability to handle complex, multi-hop questions is currently limited. Extending the meta-learning approach in the multi-episode phase to handle such questions is an important direction for our future work. Finally, the efficiency of AcKnowledge's knowledge representation could be improved by back-end storage, preferably a vector database (Zhang et al., 2023), for efficient information retrieval. In addition, we are also working on integrating moderate to no-offensive content retrieval as knowledge during the internet search by implementing Google's 'safe search' feature.

A.2 Language Model Insights

We provide the detailed architectural details of the language model developed for AcKnowledge. It is based on the standard transformer network and comprises the following components.

- **Hidden size**: The hidden size or embedding dimension is set to 512. This represents the

dimensionality of the hidden states and embeddings used in the model.

- **Intermediate Hidden Size**: The intermediate hidden size in the FeedFoward layer is set to 4 times the hidden size. In this case, it would be 4 ∗512 = 2048.
- **Context Length**: The maximum context length or block size is set to 256. This determines the maximum number of tokens that the model can process in a single sequence.
- **Heads**: The number of attention heads in the multi-head attention layer is set to 8. This allows the model to attend to different parts of the input sequence simultaneously.
- **Layers**: The model's number of transformer blocks or layers is set to 8. Each block consists of a multi-head attention layer followed by a FeedFoward layer.
- **Vocabulary Size**: The vocabulary size is not explicitly mentioned in the provided code. It would depend on the size of the vocabulary used during the model's training.
- **Token and Position Embeddings**: The input tokens are mapped to embeddings using an embedding table, and position embeddings are added to capture positional information.
- **Transformer Blocks**: The model consists of a sequence of transformer blocks, each containing a multi-head attention layer and a FeedFoward layer. The multi-head attention layer allows the model to attend to different parts of the input sequence, while the FeedFoward layer applies non-linear transformations.
- **Layer Normalization**: Layer normalization is applied after each multi-head attention and feedforward layer to normalize the activations and improve training stability.
- **Linear Output Layer**: The final hidden states are passed through a linear layer to generate temporal records or logits for each token in the vocabulary.

The total number of parameters in the model is calculated to be around 124.6 million, or approximately 125 million. In addition, we implement layer normalization instead of batch normalization for our model.

A.3 Dataset Details

We offer more thorough explanations of the datasets that we utilized in our experiments. The sources from which we obtained the datasets and the sources from which the authors originally made them available are included. For information on these sources' licenses or conditions of use and/or dissemination, we direct readers to them. To the best of our knowledge, neither objectionable content nor names or unique identifiers for specific individuals can be found in the databases that are used.

- **SQuAD 2.0**: The dataset was originally released in 2018, and was made publicly available at https://rajpurkar.github.io/SQuAD-explorer/explore/v2.0/dev/. We obtain the dataset from https://huggingface.co/datasets/rajpurkar/squad_v2.
- **NQ**: The dataset was originally released in 2019, and was made publicly available at https://ai.google.com/research/NaturalQuestions. We obtain the dataset from https://huggingface.co/datasets/natural_questions.

A.4 Benchmarking Setup

The experiments are conducted with standard computational resources for NLP and machine learning experiments. Specifically, the hardware configuration includes an Intel Xeon CPU with a 2.20 GHz clock speed and 12 GB of RAM.

We use the NVIDIA A100 GPU (Markidis et al., 2018) for our experiments. It is built on Tensor Core architecture, up to 80 GB of GPU memory, and up to 312 TFLOPS single precision performance. The parallel processing ability of the GPU makes the meta-learning and iterative learning based on user feedback of the model more robust, as well as the benchmarking process more time-efficient. This setup ensures that the models can be trained by exploiting the high computational power and memory bandwidth of the GPU.

The training durations for all SoTA and baseline models range from 8 to 15 hours on SQuAD 2.0 and NQ datasets. BLOOM 350M takes the highest time to train on both datasets, i.e., approximately 15 hours, while BERT-base takes the lowest time to train, i.e., 8 hours. AcKnowledge takes approximately 10 hours to train on SQuAD 2.0 and NQ datasets.

A.5 Extensive Evaluation

Furthermore, we showcase the performance of the compared language models on two widely adopted

benchmarks, Massive Multitask Language Understanding (MMLU) and BIG-Bench Hard (BBH). The evaluation of all the compared models is conducted in their standard settings without any additional fine-tuning, whereas AcKnowledge is evaluated in the meta-learning (*M-L*) setting as proposed.

The MMLU benchmark comprises a diverse array of NLP tasks, including question-answering, commonsense reasoning, and natural language inference, among others. In contrast, the BBH benchmark is a curated subset of challenging tasks from the BIG-Bench suite, designed to assess the capabilities of LLMs in complex and specialized instructions. Table 4 presents the benchmark results for MMLU, BBH, and the average performance of each model.

Models	**MMLU**	**BBH**	**Average**
BLOOM	42.7	28.3	35.5
OPT	43.2	27.9	35.6
ELECTRA	41.9	28.1	35.0
Flan T5-Base	44.1	28.5	36.3
DBV3-Base	43.6	28.2	35.9
GPT-Neo	40.8	27.6	34.2
MiniLM-XLMR	39.5	26.9	33.2
AcKnowledge (M-L)	**45.3**	**28.4**	**36.9**

Table 4: Performance comparison of language models on MMLU and BBH benchmarks.

Here also, AcKnowledge achieves a higher score of 45.3 on the MMLU benchmark compared to the other models, demonstrating its superior performance in language understanding on a broader scale. However, on the BBH benchmark, it obtains a score of 28.4, which is comparable to the scores of the other models, indicating similar performance.

The Average score shows that AcKnowledge has the highest overall performance at 36.9, followed by Flan T5-Base at 36.3. This suggests that the proposed framework can maintain a good balance between a strong performance on MMLU and a competitive performance on BBH.

For all the benchmarking experiments on SQuAD 2.0, NQ, MLMU, and BBH, the respective language models answer from their pre-trained knowledge. However, AcKnowledge answers with the aid of internet search and meta-learning, providing answers with the latest information and updates. Also, the user feedback in AcKnowledge for the answers is an efficient and scalable approach to continuously learning and improving the quality of answers based on real-time knowledge representation.