OpenScholar Results

Question 1

What are the main approaches for retrieval-augmented generation in large language models?

Answer

Retrieval-Augmented Generation (RAG) models combine pre-trained parametric and non-parametric memory for language generation. The main approaches for RAG include:

1. Introducing a general-purpose fine-tuning recipe for RAG models where the parametric memory is a pre-trained seq2seq model and the non-parametric memory is a dense vector index of Wikipedia, accessed with a pre-trained neural retriever [0].
2. Retrieval-Augmented Language Model Pre-training (REALM), which augments language model pre-training with a learned textual knowledge retriever trained end-to-end using a masked language modeling objective to retrieve documents that help improve the prediction of the masked tokens [1].
3. Self-RAG, which trains an LM that adaptively retrieves passages on-demand and generates and reflects on retrieved passages and its own generations using special reflection tokens [2].

These approaches aim to leverage external knowledge from large-scale text corpora to enhance the language generation capabilities of pre-trained language models.

References

[0] Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks □ - Citations: 2500 - Large pre-trained language models have been shown to store factual knowledge in their parameters, and achieve state-of-the-art results when fine-tuned on downstream NLP tasks. However, their ability t…

[1] REALM: Retrieval-Augmented Language Model Pre-Training □ - Citations: 1800 - Language model pre-training has been shown to capture a surprising amount of world knowledge, crucial for NLP tasks such as question answering.

[2] Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection □ - Citations: 450 - Despite their remarkable capabilities, large language models (LLMs) often produce responses containing factual inaccuracies due to their sole reliance on the parametric knowledge they encapsulate.

Question 2

How do transformer models handle long-range dependencies in sequences?

Answer

Transformer models can handle long-range dependencies in sequences through the use of self-attention mechanisms [0]. Self-attention allows each position in a sequence to attend to all other positions, enabling the model to capture long-range relationships between elements. In the case of the Transformer model, it relies entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution [0].

However, traditional Transformer models can struggle with handling sequences longer than a certain length due to their quadratic complexity in terms of computational cost and memory usage [1]. To address this limitation, modified Transformer architectures like Longformer have been proposed. Longformer uses a drop-in replacement for the standard self-attention mechanism that combines local windowed attention with task-motivated global attention [1], allowing it to scale linearly with the sequence length and process documents of thousands of tokens or longer.

In summary, transformer models can handle long-range dependencies through self-attention mechanisms [0]. While traditional Transformer models face limitations in handling very long sequences, modified architectures like Longformer offer a solution by scaling linearly with sequence length while maintaining the benefits of self-attention [1].

References

[0] Attention Is All You Need - Citations: 95000 - The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder.

[1] Longformer: The Long-Document Transformer - Citations: 3200 - Transformer-based models are unable to process long sequences due to their self-attention operation, which scales quadratically with the sequence length.