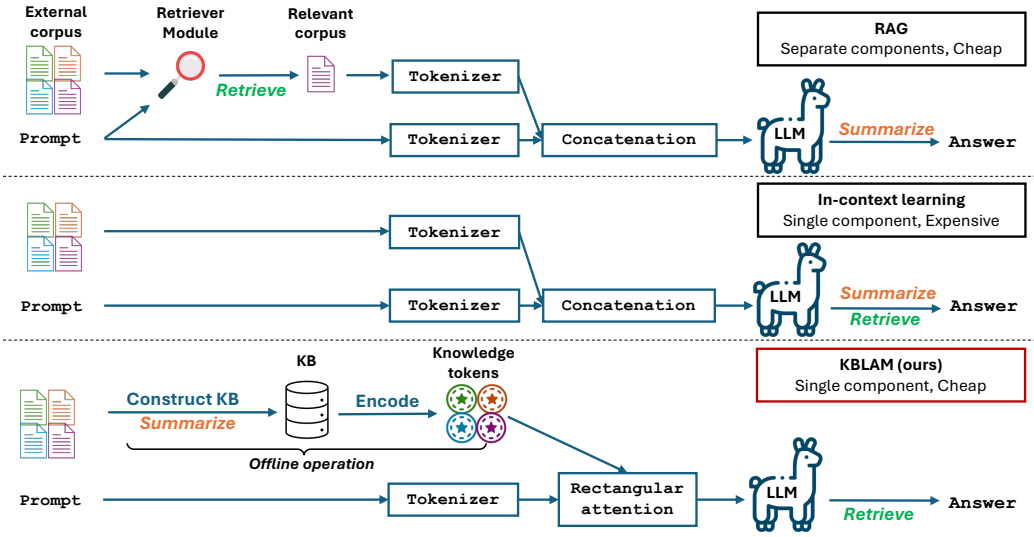

Figure 1: **Overview of the KBLaM pipeline and comparison with existing approaches.** KBLaM augments knowledge into a pre-trained LLM in the form of knowledge tokens, a set of continuous key-value vectors, using a modified rectangular attention structure. Unlike RAG, KBLaM does not rely on separate retriever module at inference time and unlike in-context learning, KBLaM's computation and memory overhead scales *linearly* rather than quadratically with the size of the KB.

demonstrations. However, the time and memory complexities of in-context learning are quadratic with the context length. Additionally, the dependencies between tokens in the context introduce difficulties in interpreting the attention matrix and dynamically updating the KV cache.

In this paper, we propose a new regime for augmenting pre-trained LLMs with external knowledge: **K**nowledge **B**ase augmented **La**nguage **M**odel (KBLaM, Fig. 1 bottom). We consider a setting where the *unstructured* external corpus is transformed into a *structured* knowledge base (KB) through existing tools, which summarizes the critical information in the data and generates knowledge triples (Eq. (1)) containing an entity name (`<name>`), a property (`<property>`), and a value (`<value>`). Then, KBLaM's design fully leverages the structure of the KB, achieving efficient integration of external knowledge into LLMs (Sec. 4).

First, KBLaM maps each triple into a *fixed-length* key-value vector pair, referred to as a *knowledge token*, which has sizes identical to the KV cache of a single token and can be seamlessly incorporated into each attention layer of the LLM. In particular, KBLaM encodes from `<name>` and `<property>` into a key vector, which serves as an identifier, mimicking the key embedding of a token; `<value>` into a value vector, which provides the actual content, similar to the value embedding of a token.

Then, KBLaM uses the structure between triples to augment knowledge tokens into the LLM's attention in a scalable and dynamic way using a simple rectangular attention structure: Triples with different `<name>` and `<property>` can be considered to represent independent pieces of information, therefore knowledge token from each triple can be encoded and injected into pre-trained LLM independently. This allows the KBLaM's complexity (memory and time) to grow linearly with respect to the number of triples, unlike in-context learning's quadratically growing overhead, giving KBLaM much better scalability. Additionally, the independence allows us to update/remove/add a triple by only modifying its corresponding single knowledge token without any further changes, which is not achievable in, e.g. standard KV cache mechanism. Perhaps more importantly, the attention matrix under this design is highly interpretable. As we show in Fig. 4, the model's use of the knowledge tokens can be directly inspected through the attention score.

Lastly, we show that the linear adapters can be learned using instruction tuning on purely synthetic data (Sec. 5) while being able to generalizes to real data. Different from supervised fine-tuning, which aims to memorize knowledge into model weights, the learning process of KBLaM aims at finding a projection between the pre-trained sentence encoder space and the LLM's embedding space,