

Design Patterns for LLM-based Neuro-Symbolic Systems

Maaike de Boer[a], Quirine Smit[a,*], Michael van Bekkum[a], André Meyer-Vitali[b] and Thomas Schmid[c,d]
[a] *Dep. Data Science, TNO, The Hague, The Netherlands*
[b] *Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), Saarbrücken, Germany*
[c] *Medical Faculty, Martin Luther University Halle-Wittenberg, Halle (Saale), Germany*
[d] *Leipzig School of Computing and Communications, Lancaster University, Leipzig, Germany*

Abstract. Large Language Models (LLMs) have been a dominating trend in Artificial Intelligence (AI) the past few years. At the same time, neuro-symbolic systems have also received increasing interest due to their advantages over purely statistical generative models. However, it is currently difficult to compare the different ways in which the training, fine-tuning and usage of the growing variety of such approaches is carried out. In this work, we use and extend the modular design patterns and Boxology language of van Bekkum et al for this purpose. These patterns provide a general language to describe, compare and understand the different architectures and methods used. The primary goal of this work is to support better understanding of LLM-based models that are used in combination with knowledge based systems, making them neuro-symbolic systems. In order to demonstrate the usefulness of this approach, we explore LLM-based neuro-symbolic architectures and approaches as well as use cases for these design patterns.

Keywords: design patterns, neuro-symbolic AI, generative models, Large Language Models

1. Introduction

In recent years, Artificial Intelligence (AI) has taken a leap and reached a level of capacity and productivity unprecedented in previous decades. In the form of so-called generative AI, and Large Language Models (LLMs) in particular, complex statistical approaches have demonstrated natural language processing capabilities at level very close to human capabilities. Prominently, the release of OpenAI's ChatGPT system[1] has changed world of text generation forever. Currently, a wealth of many different LLM models are being developed and published, both open-source and proprietary [7, 14, 28, 40]. Despite of the many impressive achievements and capabilities of LLMs, however, major open challenges of purely statistical LLMs remain, such as hallucination [23], explainability [84] and trustworthiness [22, 33].

In response to these challenges, a variety of novel neuro-symbolic approaches to LLM-based AI systems have emerged lately [11, 67]. Due to the quantity and diversity of emerging generative techniques, however, it becomes more and more challenging to keep track of the ever-growing variety of models with different architectures and capabilities. This challenge becomes even more complex with the emerging trend to combine LLMs with symbolic AI techniques. One of the solutions to tackle this issue is to apply a high-level conceptual framework to discuss,

*Corresponding authors, both contributed equally: maaike.deboer@tno.nl, quirine.smit@tno.nl.
[1]https://chat.openai.com/chat

compare, configure and combine different models by using a Boxology. The Boxology started in the field of neuro-symbolic systems, by [60] in 2019. This work is extended in 2021 by [59] by providing a taxonomically organised vocabulary to describe both processes and data structures used in hybrid systems.

In this paper, which is an extension of our previous paper [12], we propose to use and extend the Boxology to gain insights in a variety of LLMs, specifically on LLMs used in a neuro-symbolic approach. To this end, this paper provides two contributions: Firstly, we propose novel design patterns as extension of the current Boxology to promote transparency and trustworthiness in system design, by providing interpretable, high-level component descriptions of LLM-based neuro-symbolic systems. Our modular approach supports new architectures and engineering approaches to systems based on LLMs. Secondly, we test validity and usefulness of the Boxology and our extensions in this field on example architectures and applications, such as ChatGPT, KnowGL, GENOME and Logic-LM.

The rest of the paper is organized as followed. In the next section, we give a more detailed overview of the related work regarding LLMs and LLM-based neuro-symbolic systems and Boxology. In the third section, we propose to extend the Boxology by three novel basic patterns in order to be able to handle LLMs, and we explain several compositional design patterns in this field. In section 4, we dive into specific applications and tasks in which LLMs, specifically in neuro-symbolic systems, are used. We conclude with summarizing our key findings and outlining future work.

2. Related Work

2.1. LLMs and LLM-based Neuro-Symbolic Systems

The key technology most current LLMs use is the so-called transformer architecture. The original transformer architecture published by [61] proposed to use two interacting models, an encoder and a decoder. These can be trained end-to-end (such as flan-T5 [10]). Alternatively architectures have been proposed using either only the encoder-only (BERT [13]) or decoder-only (GPT [5], BLOOMZ [42], PaLM [9]) models. As only few LLMs based on other architectures have been proposed [2, 48], in this paper we focus on transformer-based LLMs and consider encoder-only, decoder-only and encoder-decoder systems to be possible types of LLMs.

The difference between encoder-only and decoder-only systems is motivated by optimizing the architecture for different scenarios. Encoder-only transformers, such as BERT [13], are specialised in contextual encoding, often named base models. They use the context to encode input sentences and represent it as a machine interpretable representation, such as a vector representation. Decoder-only systems are complementary to the encoder-only paradigm, but structurally different [39]. A decoder-only system decodes the input data directly, without being transformed into a higher, more abstract representation, to the desired representation (text, images or otherwise). Examples of this are generative models from the GPT family [5]. Decoder-only architectures can be further divided into causal decoder architectures and prefix decoder architectures. Causal decoder architectures, such as GPT [5, 50] and BLOOMZ [42], use only unidirectional attention to the input sequence by using a specific mask. Prefix decoder architectures, such as PaLM [9], use the bidirectional attention for tokens in the prefix while maintaining unidirectional attention for generating subsequent tokens.

Despite many impressive capabilities and results in challenging benchmarks, purely statistical LLM-based systems continue to exhibit unwanted side effects, such as hallucinations and lack of explainability. Combinations of symbolic AI and machine learning are already extensively used in the area of Natural Language Processing [46]. While originally, the main application area was natural language understanding (e.g. text classification, sequence labeling, and question answering), newer hybrid NLP applications focus on natural language generation and natural language reasoning (e.g., language modeling, dialogue systems, text summarization, machine translation, question generation). One paper that provides an overview of different approaches in the field of combining LLMs and symbolic systems is [45]. In this overview, a distinction is made between KG-enhanced LLMs, LLM-augmented KGs and synergized LLMs + KGs. For KG-enhanced LLMs, two primary approaches have been explored: incorporation during the pre-training stage to facilitate knowledge acquisition, and utilization during the inference stage to improve access to domain-specific information. Additionally, KGs have been employed post-hoc to augment the interpretability of LLMs, elucidating both factual content and reasoning processes. In order to augment KGs, LLMs have been employed as text encoders to enrich KG representations and extract relations and entities from original corpora. Recent studies have focused on designing KG prompts that effectively convert structural KGs into LLM-comprehensible formats, enabling direct application of LLMs to KG-related tasks such as completion and reasoning. Moreover, it has been proposed by the authors to consider effects and concepts of syngerized LLMs + KGs with respect to four layers: 1) Data, 2) Synergized Model, 3) Technique, and 4) Application. We will loosely use the categorisation of this paper in our exploration of different LLM-based neuro-symbolic systems.

2.2. *Boxology*

We will base our paper on the previous work of van Bekkum and colleagues [59], in which a taxonomically organised vocabulary is provided to describe both processes and data structures used in hybrid AI systems. Elementary patterns of this approach are displayed in Fig. 1.

The highest level of this taxonomy contains instances, models, processes and actors, which may be described as follows:

Instances: The two main classes of instances are data and symbols. *Symbols* are defined as to have a designation to an object, class or a relation in the world, which can be either atomic or complex, and when a new symbol is created from another symbol and a system of operations, it should have a designation. Examples of symbols are labels (short descriptions), relations (connections between data items, such as triples) and traces (records of data and events). *Data* is defined as not symbolic. Examples are numbers, texts, tensors or streams.

Models: Models are descriptions of entities and their relationships, which can be statistical or semantic. *Statistical* models represent dependencies between statistical variables, such as LLMs or Bayesian Networks. *Semantic* models specify concepts, attributes and relationships to represent the implicit meaning of symbols, such as ontologies, taxonomies, knowledge graphs or rule bases.

Processes: Processes are operations instances and models. Three types of processes are defined: generation, transformation and inference. *Generation* can be done using, for example, the training of a model or by knowledge engineering. *Transformation* is the transformation of data, for example from knowledge graph to vector space. *Inference* can be inductive or deductive, in which induction generalises instances and deduction reaches conclusions on specific instances, such as with classification.

Actors: Actors can be humans, (software) agents or robots (physically embedded agents). [**?**] extended the original paper with a definition of teams of actors in the Boxology.

Besides the vocabulary, the visual language is defined in [59], as an extension of [60]. The visual language consists of rectangular boxes (instances), hexagonal boxes (models), ovals (processes) and triangles (actors) and untyped arrows between them. Within the boxes the concept will be noted by each level in the vocabulary using colon-separation from most generic to most-specific, for example a neural network will be model:stat:NN.

[59] present elementary patterns, which can then be combined into more complex patterns. Patterns 1a and 2a from Figure 1, for example, can be combined into a pattern which is named 3a in the paper (depicted in Figure 2). Whereas 1a describes the pattern of training a model based on data (data generates a model), 2a describes the usage of the model in deducing a symbol (data and model deduce a symbol), such as a prediction. The combination in 3a describes a basic structure for a (statistical) Machine Learning (ML) model depicting the training (creating the model) and testing or application phase (applying the model on new data).

In the past years, the Boxology has been used and extended in different ways. Three of the papers are the formalisation of the notions from the Boxology and implementation in the heterogeneous tool set (Hets) [41], the extension of the Boxology for (teams of) actors [38] and the systematic study of nearly 500 papers published in the past decade in the area of Semantic Web Machine Learning [4].

We also acknowledge the ontological visual framework utilizing semantically-enhanced symbols to represent AI system components and architectures [15]. This EASY-AI framework aims to provide a standardized symbolic language for conveying the structure, purpose, and characteristics of AI systems. The approach presents the logical formalisms underpinning this visual framework, with the objective of enhancing the comprehensibility and understandability of AI system behaviors. Recently, this framework is also provided with an initial implementation named

SNOOP-AI [16]. This framework and implementation could be used in the implementation of the design patterns, as it can provide a formal conceptual foundation for the design patterns that allows formal reasoning over (compositions of) its elements. To the best of our knowledge, specific LLM-based use cases have not been tried using the formalisation and implementation yet.

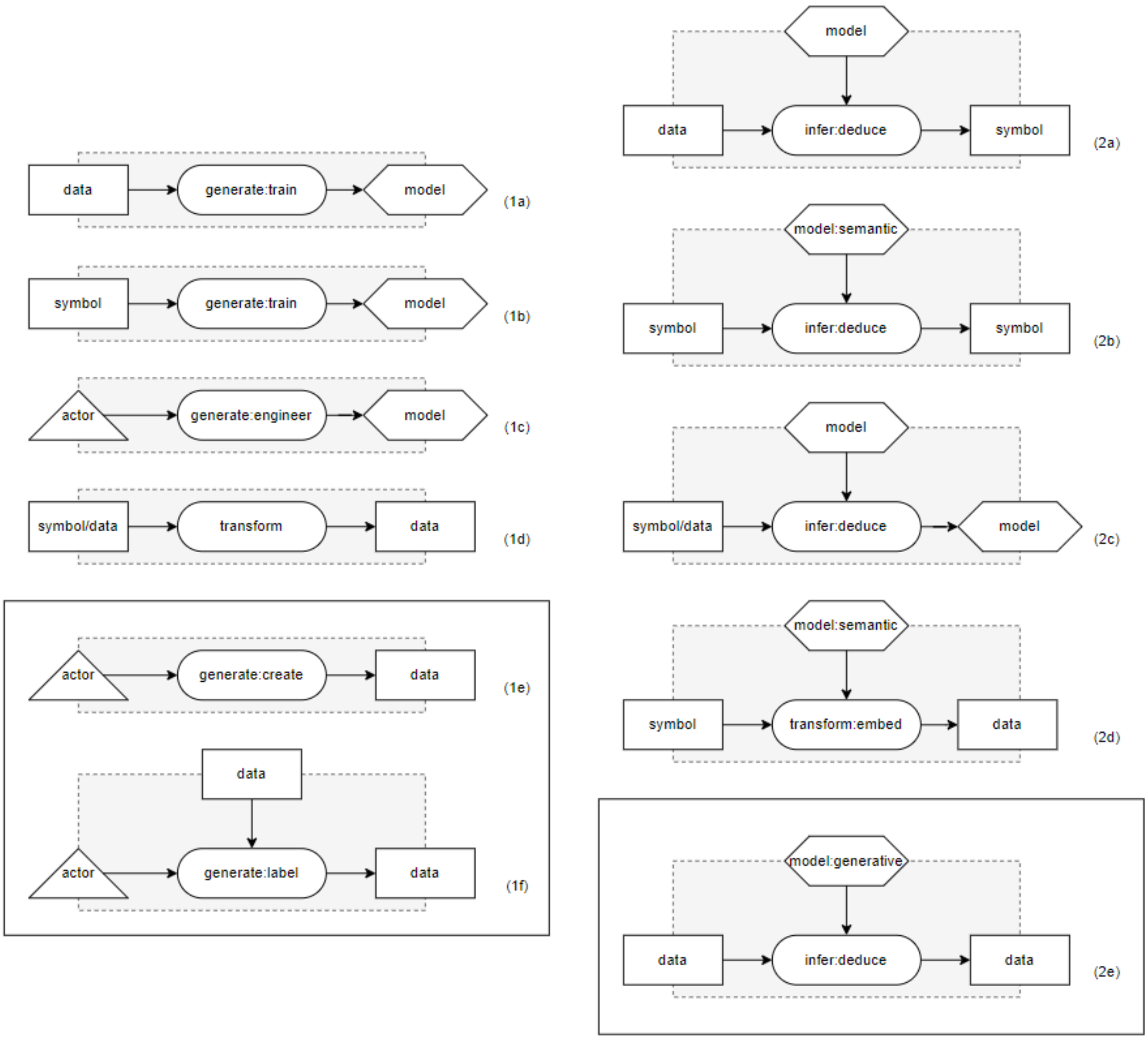

Fig. 1. All elementary design patterns, including proposed additions 1e, 1f and 2e

3. Design Patterns for LLM-based Neuro-Symbolic Systems

In this section, we propose new elementary patterns necessary to represent LLMs and LLM-based neuro-symbolic systems. We continue with an explanation of transformer models and their compositional patterns, and we finish the section with the compositional design patterns for LLM-based neuro-symbolic systems.

3.1. Introducing Novel Elementary Patterns

In order to allow for a coherent description of the Large Language Model paradigm, we propose to extend the elementary patterns of [59] that describe the generic pattern for instances, models, processes and actors (Figure 1

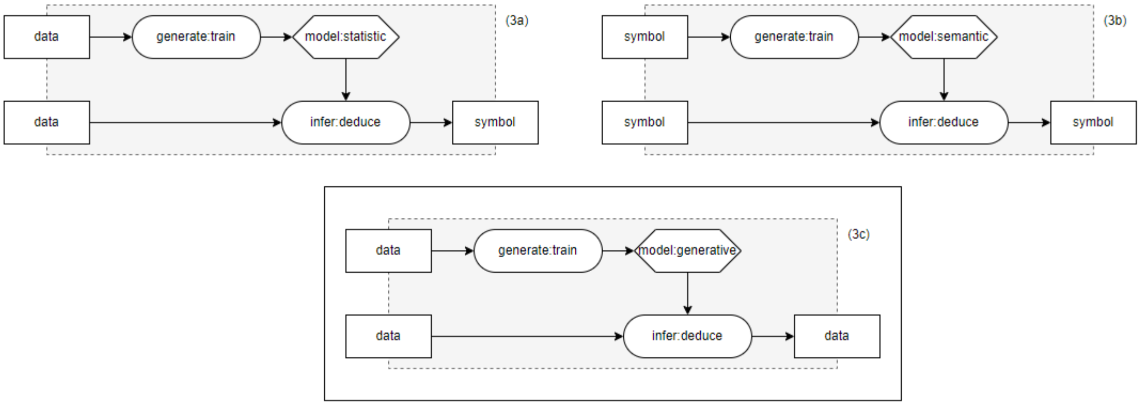

Fig. 2. Compositional design patterns, including proposed addition 3c made by combining elementary pattern 1a and 2e.

1a-1d and 2a-d). Please note that while patterns 1e and 1f are required for certain aspects of the LLM paradigm, their usage is not limited to this. Both patterns may also represent data generation and labelling by humans and may thus be employed to work with any statistical approach.

In particular, when describing classical machine learning systems, mostly pattern 2a is used, where the output is a symbol, such as a classification or a label. However, the key difference with LLMs, specifically the generative decoder models, is that the output is not a symbol, but data; this can be an image, video or text, depending on the model. Additionally, actors play an important role in LLMs, by creating prompts or label data. To this end, we here propose three new elementary patterns: pattern 1e, in which an actor can generate data, pattern 1f, in which an actor labels data, and 2e, in which a model can deduce data from data. Please note, however, that the patterns proposed in this section are transferable to other data types, for example to vision transformers, which follow a similar architecture paradigm as transformers but operate on image data.

3.2. Transformer Models

Transformers consist of an encoder and a decoder component [61]. From this basic premise, new encoder-only and decoder-only systems have been developed [5, 13]. Figure 3 shows the architecture of transformer models in Boxology. Firstly, in Figure 3A, the full transformers architecture with both encoder and decoder modules is represented. In this case we chose to accentuate the encoder and decoder as separate modules. This pattern includes the new addition compositional pattern 3c, made up of parts 1a and 2e. The decoder is a specification of a generative model introduced in 3c, whereas the encoder is a specification of a model that can be trained with data (1a).

Secondly, the use of an encoder model is shown in Figure 3B. An encoder is trained using data, Boxology pattern 1a. It is often connected to other systems, such as a classification system, pattern 3a, to be useful for tasks such as sentiment analysis. An example of this is BERT [13]. This specific pattern with classification is not considered generative, as generative models output data and not a symbol.

Thirdly, decoder-only systems are represented in Boxology as shown in Figure 3C, which is the introduced for generative models 3c. Both causal decoder architectures and prefix decoder architectures follow the same Boxology pattern.

3.2.1. Prompts and Instructions

One of the main differences between newer, generative LLMs and earlier BERT or other transformer models is that the model is fine-tuned on instructions [39]. Multi-task fine-tuning or instruction tuning, is currently often done using a collection of datasets phrased as instructions, to improve model performance and generalisation to unseen tasks [10, 79]. The original model is often referred to as foundation model [3], whereas the fine-tuned model is an adjusted model. Instruction tuning also follows pattern 1a, but this data is different as it also contains instructions.

Besides instruction learning, LLMs can also be tweaked by in-context learning. Here examples are used as part of the prompt to give context for the answers to the instructions. In this case the model weights are not changed. This

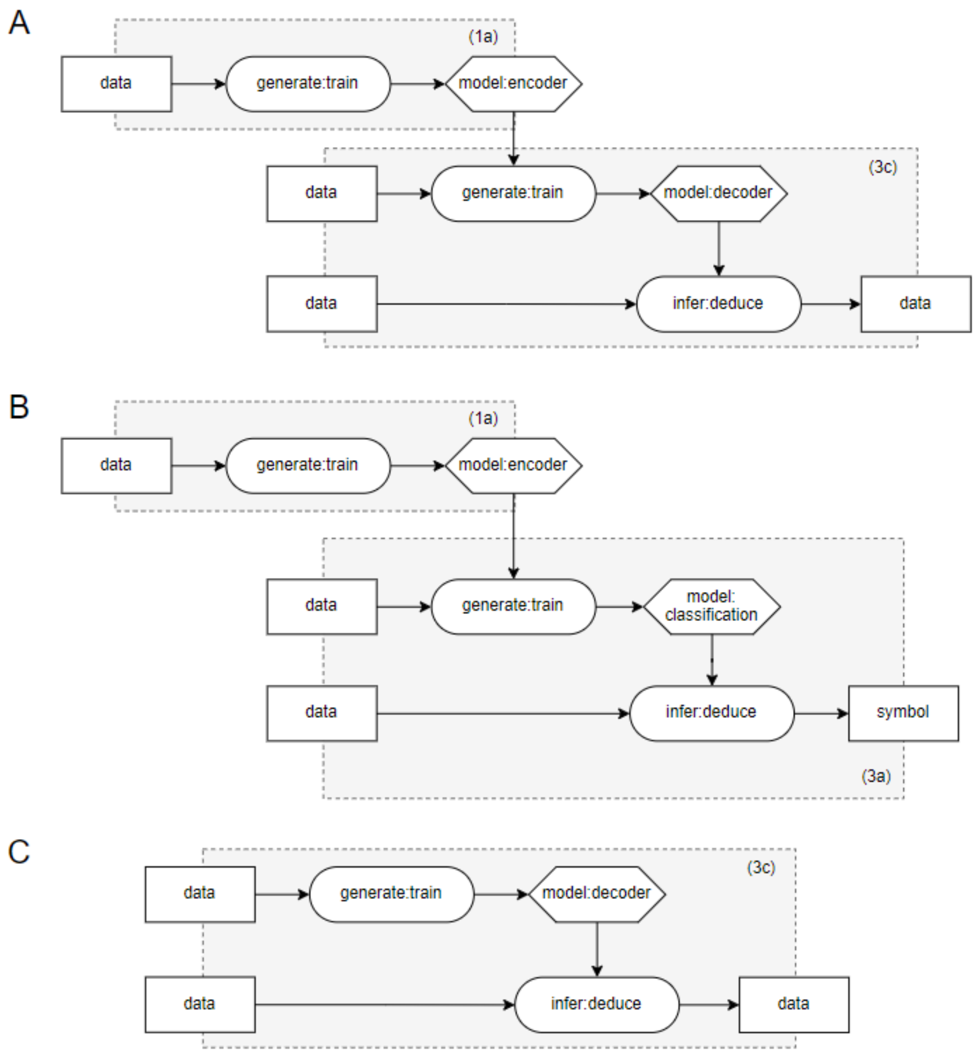

Fig. 3. Three uses of transformers. A shows the traditional encoder-decoder architecture. B shows an encoder-only model applied to a classification task. C shows a decoder-only architecture.

optimizes the performance of models on different tasks [34], but does not need as much training data as training a model from scratch. These prompts can include a few (training) examples of the input and output (few-shot) or no examples (zero-shot). These few-shot examples do not train the foundational or instruction model, and therefore we model them as input data that is used to deduce data (text), which is pattern 2e. Assistants, agents or GPTs could, however, be seen as a new model, especially if they perform other tasks, such as Retrieval Augmented Generation (RAG).

3.3. LLM-based Neuro-Symbolic Design Patterns

Even though generative LLMs generate text in convincing quality, the accuracy of their answers is lacking in many documented cases [69]. They lack in regard to facts and reasons, because they fail to understand concepts of truth, causality, time and space, as well as understanding of other physical and social relationships. This is one of the reasons to create LLM-based neuro-symbolic systems. These systems often either use an LLM followed by a semantic model, or a semantic model followed by an LLM, or a combination of two models in parallel of which the output is fused. In this section, we propose compositional design patterns for these different types of systems. We loosely follow the categorisation of [45]. We divide the section into training and application phase, as the patterns for these phases are distinct. As we only consider LLMs in these patterns, we use LLM as a type of model, and do not specify for example encoder and decoder types, as encoder-decoder, encoder-only or decoder-only models could be used in the compositional patterns.

3.4. LLM-based Neuro-Symbolic Design Patterns in Training

Generative Neuro-Symbolic Systems can use semantic models in the training of an LLM or use an LLM to create a semantic model, or can be used in synergy to create a model. In the following subsections, we will describe the different patterns in more detail.

3.4.1. KG-enhanced LLMs

The design pattern in Figure 4 shows how a semantic model (KG) can be used to transform symbols (pattern 2d) into a different type of symbol, to data (pattern 1d) that is used to train a generative model (1a). Pattern 2d is slightly adjusted, as there symbol is used as output and a general transform (not necessarily embed) is used. The constraints can represent aspects of the KG, such as semantic distance (number of hops between concepts) [53], difference in loss values between tokens and KG entities [80] or sentiment (SKEP [58]).

The data that influences the training process can, for example, act as a mask to filter the training data [30, 51, 53, 72]. In this way, the input text would be verified with respect to known entities and, therefore, increase the reliability of the training and input.

3.4.2. LLM-augmented KGs

Figure 5 shows the design pattern for the training of LLM-augmented KGs. Similar to the KG-enhanced LLMs in training, data (or symbols such as triples transformed to data) is transformed to another type of data (2d - with data as input), such as embeddings, and then used to train a (KG in this case) model (1a). The first step of using an LLM to transform the data is often used because KGs might be incomplete and textual information is not integrated in the embedding itself. For example, [43] generates representations on different levels such as sentence and document using LLMs and [21] creates multi-modal embeddings. Tasks such as LLM-augmented KG completion and construction, including Named Entity Recognition, Coreference Resolution and Relation Extraction could follow this pattern, depending on the specific implementation and whether the LLM is used in the training phase or only the application phase. For instance, KG-BERT, MLT-KGC and PKGC use LLMs for KG completion [27, 37, 75]. They use the LLM output to predict the relation between new entities and existing ones. [73] uses LLMs to aid in Named Entity Recognition, [6, 25] for Coreference Resolution and [47, 54] for Relation Extraction.

3.4.3. Synergized LLMs and KGs

One of the ways in which LLMs and KGs are synergized in training is using an LLM for joint text and KG embedding or representation. Figure 6 shows the Boxology representation of these type of systems. The symbolic triples are transformed to text (data; 1d), which is then combined with other text to incorporate both the graph structure and the textual information into the embedding simultaneously and trained to create a model (1a).

For example, kNN-KGE sees the entities as special tokens and incorporates them in the sentences as input for the LLM [63]. LMKE has a similar system structure but applies a different learning method to improve the learnt embeddings [65]. LambdaKG improves the representation of the graph structure by including neighbouring entities in the input sentence [71]. KEPLER, JointGT and DRAGON use a unified model for the knowledge embedding and pre-trained language representation [26, 64, 77]. They have pre-training tasks to come to a joint knowledge embedding and language modeling optimization. ERNIE proposes a dual encoder system, consisting of a textual encoder which is fused with the knowledge graph encoder [82]. BERT-MK has a similar dual encoder, but adds additional information from neighbouring entities in the knowledge graph [19]. Coke-BERT further improves on this idea by adding a module to filter out irrelevant neighbouring entities [55]. JAKET fuses the entity representation in the middle layers of the LLM [78].

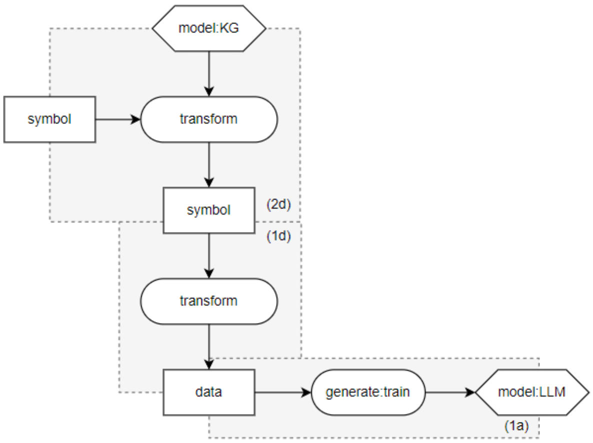

Fig. 4. KG-enhanced LLMs training.

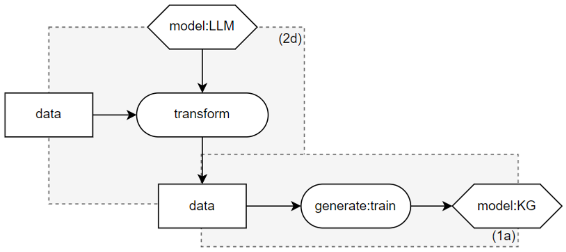

Fig. 5. LLM-augmented KGs in training

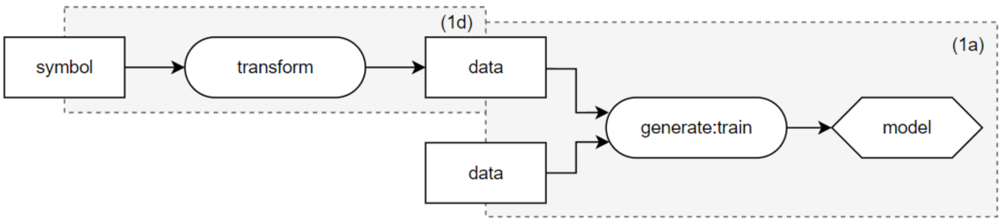

Fig. 6. Synergized LLMs and KGs in training

3.5. LLM-based Neuro-Symbolic Design Patterns in Application

Neuro-Symbolic systems often combine KGs and LLMs in the application phase. In this way, the system is more robust to new situations. Many of the LLM-based neuro-symbolic systems follow one of the pre-defined patterns. For example, if KG construction is only done using a pre-trained generative model, this is captured in pattern 2a (with data infer a symbol - the KG - using a model). The LLM-augmented KG-to-text generation can be done using the basic pattern 2d, in which a KG is inferred using a generative model, creating data (text).

3.5.1. KG-enhanced LLMs

The design pattern in Figure 7 shows how the knowledge of knowledge graphs (KG) can be included in the inferencing of LLMs. The input data can be aligned with the knowledge or augmented by adding relevant facts for the LLM to improve the output.

In contrast to the injection of KGs during training (see subsubsection 3.4.1), the pattern 2d and 1d are now input to the infer process instead of the train process. This means that the knowledge is up to date at the time of application, rather than at the time of training (which may happen a long time before).

This pattern transforms input data by aligning it with knowledge from the KG before they are fed into the deduction process with an LLM model. It could be done in a process of prompt engineering using the KG [31, 36, 62, 68] or retrieval-augmented knowledge methods such as RAG [29]. One specific architecture is KagNet, which first encodes the input KG and then augments it with textual representation [32].

3.5.2. LLM-Augmented KGs

The design pattern in Figure 8 shows how an LLM can used to deduce data from a KG. Similar to the KG-enhanced LLMs for application, the difference between training and application for LLM-augmented KGs is that the first pattern is input to infer process rather than the train process. One example is by using LLMs for KG

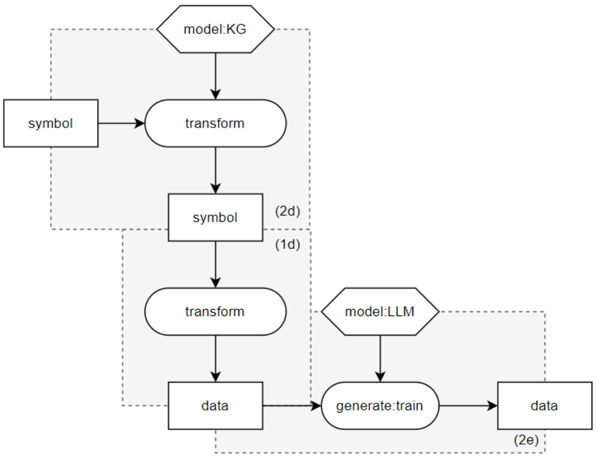

Fig. 7. KG-enhanced LLMs in application

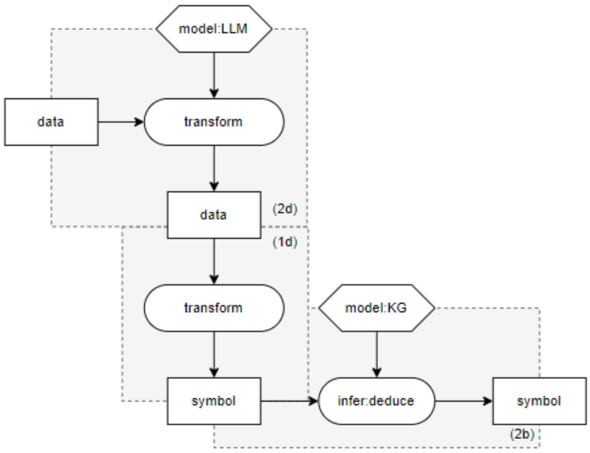

Fig. 8. LLM-augmented KGs in application

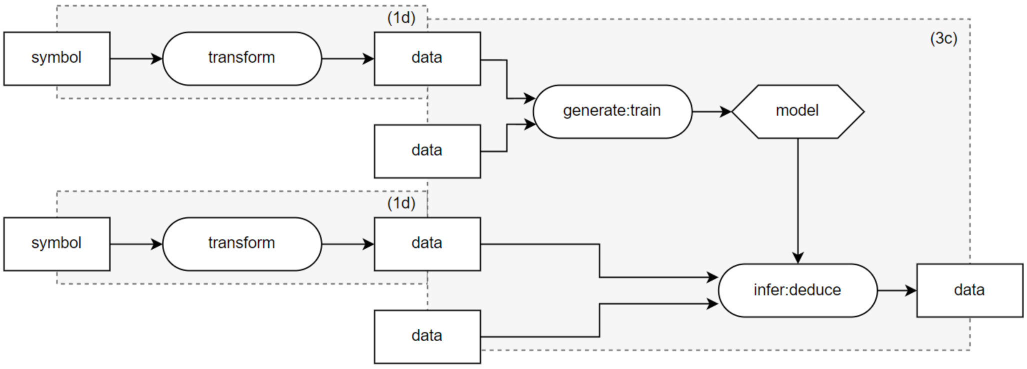

Fig. 9. Synergized LLMs and KGs for reasoning

Embedding. Pretrain-KGE uses an LLM to encode the text of the parts of the triples, and uses that encoding as a starting point for the KG encoding [83].

Moreover, in LLM-augmented KG question answering, LLMs are used to bridge the gap between natural language questions and the retrieval of answers from KGs [20, 35]. Also LLMs can be used for generation of text from a knowledge graph, where LLMs are employed to generate natural language that describes the facts from KGs [17, 56, 66]. MHGRN uses the LLM representation of the text to guide the reasoning process in the KGs [18].

3.5.3. Synergized LLMs and KGs

LLMs and KGs can be combined to work in synergy, also in the application phase. Figure 9 shows how this can be applied, specifically in the case of synergized reasoning. Examples of such methods are JointLK and GreaseLM. They have an interaction between the tokens in the textual input and the entities in the graph in the layers of the model [57, 81]. QA-GNN represents the LLM information as a special node in the KG for reasoning [76].

4. Use Cases

In this section, we describe and explore several papers that propose LLM-based neuro-symbolic system. The selected papers are chosen, as they represent a diverse set of possibilities to use an LLM, at the start of the system, in the middle and at the end, but also to act as a fluent language interface or a formal language interface. We also included ChatGPT, which is the most famous generative AI system, and although mainly data driven, includes a symbolic component in the reward modelling part of the training phase.

4.1. ChatGPT

ChatGPT is an application of the foundational model GPT3 [5], and later GPT4 [70]. It is trained further to be of aid in the setting of an assistant. The architecture of the training phases is represented in Figure 10. The foundational model GPT3 is used as a basis for further training (1a). Instructions and answers are used to train what will become ChatGPT. Then, based on new prompts the model generates a response (3c).

To further train ChatGPT to give the desired responses the reward model is added. The reward model is a separate model, which can judge if a response is a good one, given the instructions. The reward model is trained by people annotating the multiple answers to instructions. To train the reward model, the model trained on instructions is asked to output multiple answers. These answers are then ranked by annotators to generate a training set for the reward model (1f). The reward model is trained to compare answers of ChatGPT and return their score (3a). This is then used in a loop with the ChatGPT to improve the instruction answering process. As one can view, we have adapted Boxology patterns to be able to accept multiple inputs.

When applying ChatGPT in a pipeline, it suffices to show only pattern 3c, the block containing ChatGPT and 1e to show the user writing the prompt.

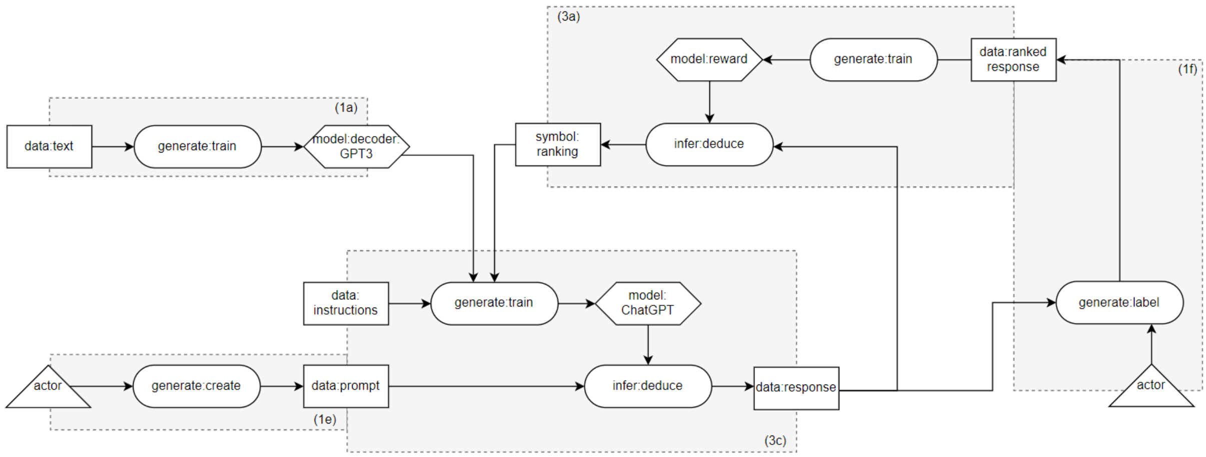

Fig. 10. Training phase of ChatGPT

4.2. RAG

Retrieval-Augmented Generation (RAG) is a method which expands an LLM with external knowledge [29]. A RAG system has two main components, a retriever and a generator. Figure 11 shows the Boxology representation of a RAG system, clearly showing the retriever and the generator. Firstly, the retriever selects relevant documents based on the posed question (2a), through classification or otherwise. Then, the question and the retrieved documents are presented to an LLM in a prompt (2e). The LLM then generates an answer to the question based on the information in the selected documents. The LLM can then also present the source of the information, which makes it more trustworthy and reliable.

In KD-CoT, KSL and Think-on-graph facts are retrieved from a KG, together with the reasoning. Then an LLM then generates a natural language answer to present to the user [17, 56, 66].

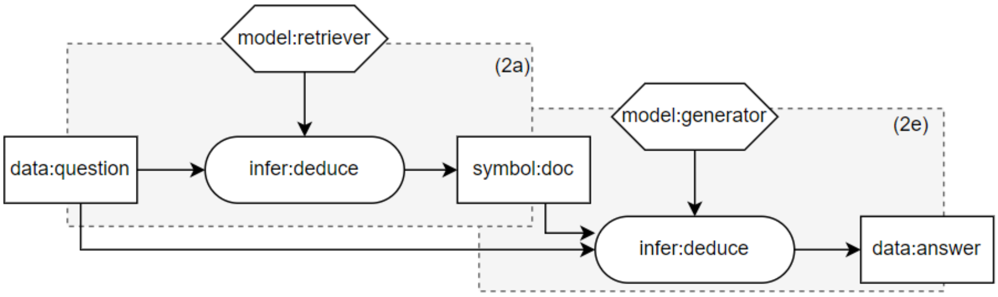

Fig. 11. Use of Retrieval-Augmented Generation

4.3. KnowGL

Figure 12 shows KnowGL Parser [52], a NeSy system combining an LLM module and symbolic methods. The KnowGL Parser can be used to automatically extract knowledge graphs from collections of documents. It is based on BART-large, which has an encoder-decoder architecture. The encoder receives a sentence (1a) and the decoder generates a list of 'subject, relation, object' (3c). These are then parsed (transformed) in preparation of the next step, fact ranking (1d). Here a ranked list is created of distinct facts and their scores (2b). In the final step the generated facts are linked to Wikidata. This is done using a mapping of labels to Wikidata IDs (2b). In the case that the generative model has created a new entity, type or relation label that are not in Wikidata it returns 'null'.

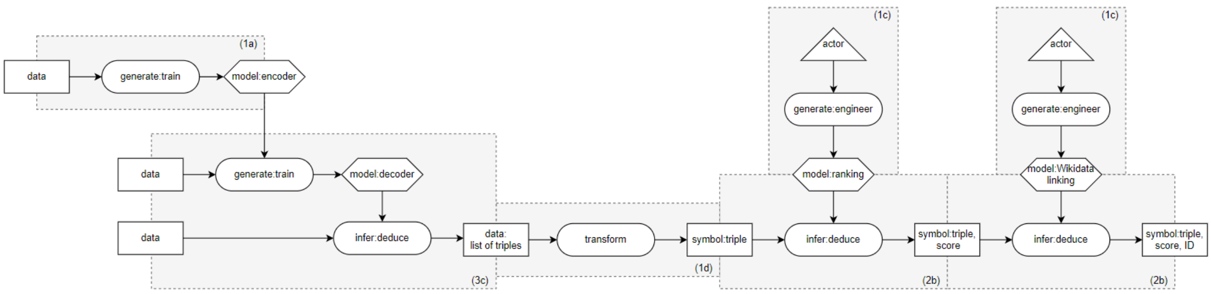

Fig. 12. Boxology representation of KnowGL [52]

4.4. KnowBERT

While knowledge is mostly injected to statistical generative models either during the input or during the output stage, also approaches to inject knowledge inside the model have been proposed. A prominent example is KnowBERT, a modified variant of the transformer architecture BERT [49]. It stands out for its fusion of contextual and graph representations, attention-enhanced entity spanned knowledge infusion, and flexibility in injecting multiple Knowledge Graphs at various model levels. By integrating so-called Knowledge Attention and Recontextualization (KAR) layers [1], graph entity embeddings are utilized that are processed through an attention mechanism to enhance entity span embeddings. This happens in later layers of the model to stabilize training but may potentially also used to inject knowledge at earlier stages [11]. The Boxology pattern for KnowBERT is depicted in Figure 13.

4.5. Mathematical Conjecturing and LLMs

The system proposed by [24] assigns the generative task of discovery of mathematical conjectures to an LLM (3c), while the results can be checked afterwards using a symbolic theorem prover or counter-example finder (2b). The

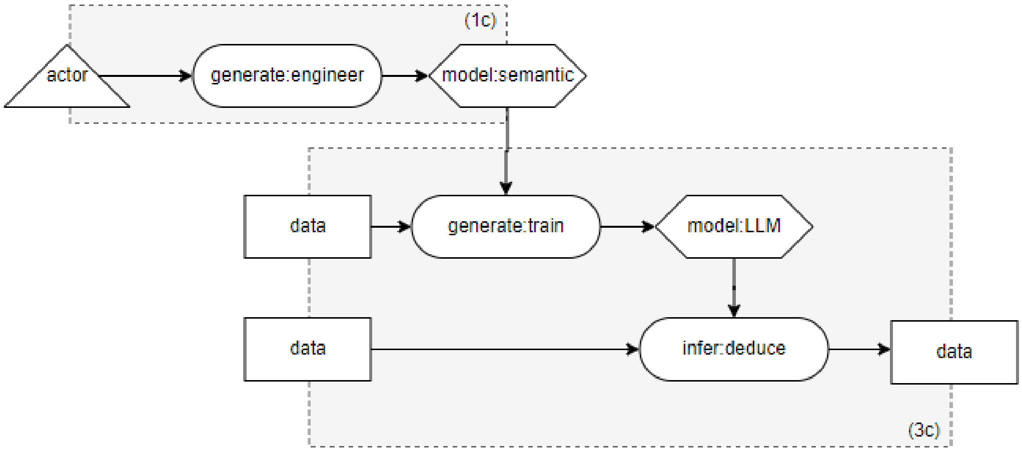

Fig. 13. Boxology representation of KnowBERT [49]

system is prompted with a formal theory (e.g. a sort function), and has the LLM generate lemmas from the theory. These generated lemmas are transformed from data to symbol and can then be used by the semantic model(s). The pattern is depicted in Figure 14. The approach taken in [74] is also captured by this representation. The system proposed uses an LLM component to produce Prolog code (3c) and a symbolic inference engine to produce answers and reasoning traces by executing the aforementioned code (1d, 2b).

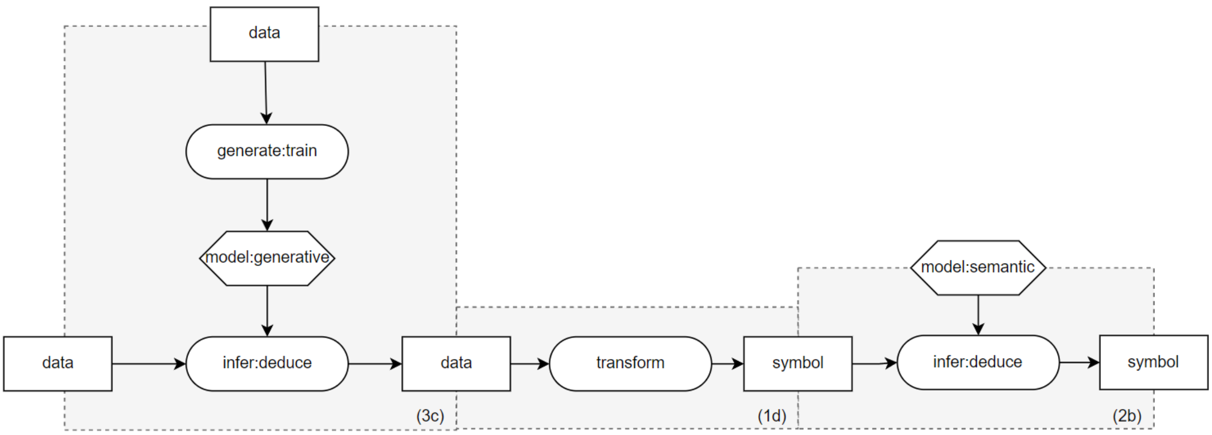

Fig. 14. Boxology representation of using LLMs for discovery of mathematical conjectures [24]

4.6. GENOME

Generative Neuro-Symbolic Visual Reasoning by Growing and Reusing Modules (GENOME) [8] focuses on the task of generative software module learning, based on an LLM generating signatures (input/output) and reasoning steps, then have an LLM create the software module based on those and evaluate the module on test cases.

The system consists of three stages: module initialization, module generation, and module execution. The representation for this paper is depicted in Figure 15. First an LLM assesses a visual-language question and outputs new module signatures and operation steps as a response to the query (3c), if current modules cannot provide an adequate response. In the next step, the LLM creates a module (software code) based on the signature/test case (3c). Finally the module is executed by passing it a visual query (2a).

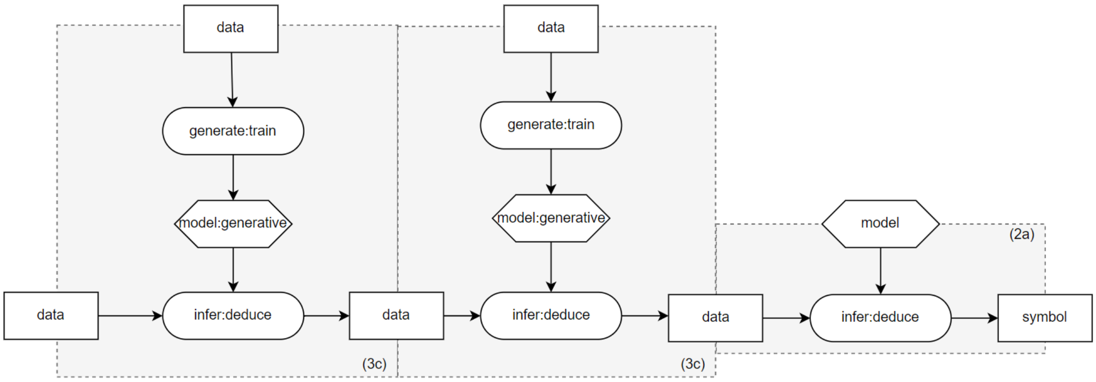

Fig. 15. Boxology representation of GENOME [8]

4.7. *Logic-LM*

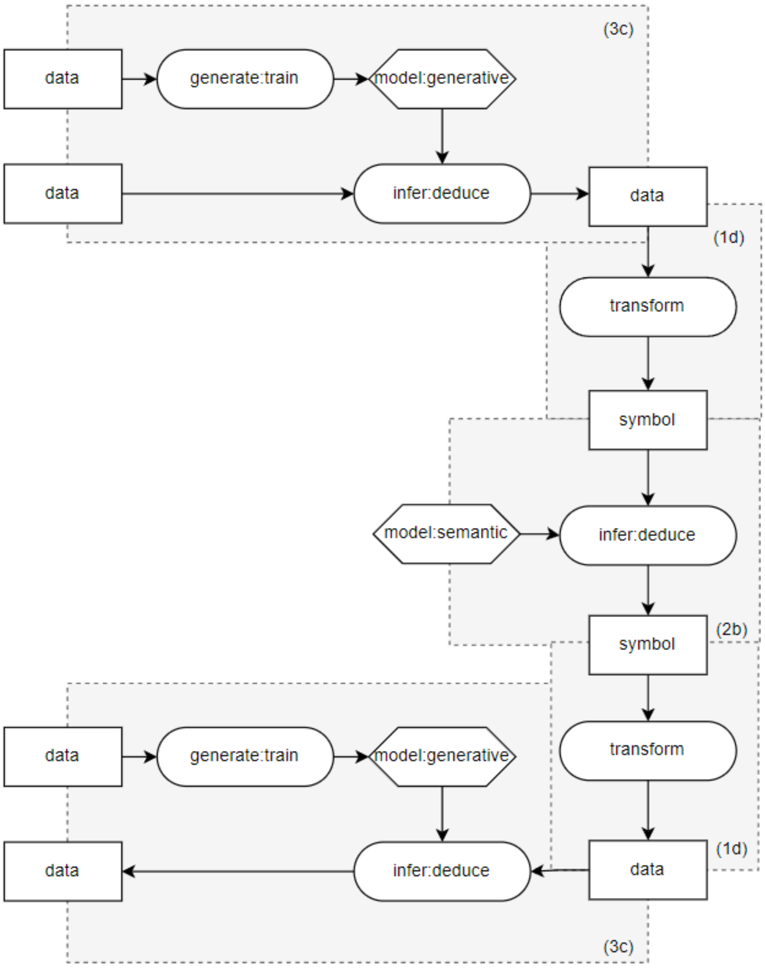

Fig. 16. Boxology representation of Logic-LM [44]

Logic-LM [44] integrates LLMs with symbolic solvers to improve logical problem-solving. This paper is depicted in Figure 16: the system utilizes LLMs to translate a problem stated in natural language problem into a symbolic formulation (3c). In the next step, a symbolic reasoner performs logical inference on the formulated problem (1d, 2b, 1d). Finally, an LLM interprets the results and outputs natural language (3c). The LLM thus functions as a fluent language interface (both on input and output) to a symbolic reasoner component.

5. Conclusion and Future Work

LLMs are currently often used in many diverse applications. Combining data-driven approaches with knowledge-based techniques is a promising development to this end. In this paper, we propose new design patterns for modular LLM-based neuro-symbolic systems to be included into the design pattern approach for neuro-symbolic systems as proposed by [59]. We show how the composition of elementary patterns can be used to describe LLMs, and we explore several categories as well as specific approaches in use cases, such as ChatGPT, KnowGL, GENOME and Logic-LM.

In future work, we expect to further extend this work towards adjacent domains, such as generative AI systems in general or multi-modal generative AI systems. In addition we expect to further extend and deepen the Boxology itself. For example, temporal or recurring/iterative aspects are not yet taken into account and cannot be visualised well. Current investigation has also shown that concept naming and labelling and formalisation of the Boxology needs revisiting. Then there is the do's and don'ts: the extension has raised questions about which pattern combinations are allowed and which are not. The importance of modelling datasets for LLMs or generative AI in general may be taken into account in future specifications of particular subtypes of Instances and Models in the taxonomy. Additionally, the use of graphical tools for software development is well-known from the Unified Modelling Language (UML) and visual programming tools, such as LabView or Scratch. We are mostly concerned with graphical representations of design patterns for system design and documentation, but the promise of templates, low-code or no-code development is appealing for the future.

Acknowledgements

We would like to thank the TNO project GRAIL for their financial support, as well as Frank van Harmelen and Annette ten Teije for their feedback. We would also like to thank Daan Di Scala for his contribution to the KnowGL pattern.

References

[1] I. Balažević, C. Allen and T.M. Hospedales, Tucker: Tensor factorization for knowledge graph completion, *arXiv:1901.09590* (2019).

[2] M. Beck, K. Pöppel, M. Spanring, A. Auer, O. Prudnikova, M. Kopp, G. Klambauer, J. Brandstetter and S. Hochreiter, xLSTM: Extended Long Short-Term Memory, *arXiv preprint arXiv:2405.04517* (2024).

[3] R. Bommasani, D.A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M.S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill et al., On the opportunities and risks of foundation models, *arXiv:2108.07258* (2021).

[4] A. Breit, L. Waltersdorfer, F.J. Ekaputra, M. Sabou, A. Ekelhart, A. Iana, H. Paulheim, J. Portisch, A. Revenko, A.t. Teije et al., Combining machine learning and semantic web: A systematic mapping study, *ACM Computing Surveys* (2023).

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J.D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell et al., Language models are few-shot learners, *Advances in neural information processing systems* **33** (2020), 1877–1901.

[6] A. Cattan, A. Eirew, G. Stanovsky, M. Joshi and I. Dagan, Cross-document Coreference Resolution over Predicted Mentions, in: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 5100–5107. doi:10.18653/v1/2021.findings-acl.453. https://aclanthology.org/2021.findings-acl.453.

[7] H. Chen, F. Jiao, X. Li, C. Qin, M. Ravaut, R. Zhao, C. Xiong and S. Joty, ChatGPT's One-year Anniversary: Are Open-Source Large Language Models Catching up?, *arXiv:2311.16989* (2023).

[8] Z. Chen, R. Sun, W. Liu, Y. Hong and C. Gan, Genome: generative neuro-symbolic visual reasoning by growing and reusing modules, *arXiv preprint arXiv:2311.04901* (2023).

[9] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H.W. Chung, C. Sutton, S. Gehrmann et al., Palm: Scaling language modeling with pathways, *Journal of Machine Learning Research* **24**(240) (2023), 1–113.

[10] H.W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma et al., Scaling instruction-finetuned language models, *arXiv:2210.11416* (2022).

[11] P. Colon-Hernandez, C. Havasi, J. Alonso, M. Huggins and C. Breazeal, Combining pre-trained language models and structured knowledge, *arXiv preprint arXiv:2101.12294* (2021).

[12] M. de Boer, Q. Smit, M. van Bekkum, A. Meyer-Vitali and T. Schmid, Modular Design Patterns for Generative Neuro-Symbolic Systems, *GeNeSy* (2024).

[13] J. Devlin, Bert: Pre-training of deep bidirectional transformers for language understanding, *arXiv preprint arXiv:1810.04805* (2018).

[14] A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Yang, A. Fan et al., The llama 3 herd of models, *arXiv preprint arXiv:2407.21783* (2024).

[15] A. Ellis, B. Dave, H. Salehi, S. Ganapathy and C. Shimizu, EASY-AI: sEmantic And compoSable glYphs for representing AI systems, in: *HHAI 2024: Hybrid Human AI Systems for the Social Good*, IOS Press, 2024, pp. 105–113.

[16] A. Ellis, B. Dave, H. Salehi, S. Ganapathy and C. Shimizu, Implementing SNOOP-AI in CoModIDE, in: *NAECON 2024-IEEE National Aerospace and Electronics Conference*, IEEE, 2024, pp. 101–104.

[17] C. Feng, X. Zhang and Z. Fei, Knowledge solver: Teaching llms to search for domain knowledge from knowledge graphs, *arXiv preprint arXiv:2309.03118* (2023).

[18] Y. Feng, X. Chen, B.Y. Lin, P. Wang, J. Yan and X. Ren, Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering, in: *Proceedings of EMNLP*, Association for Computational Linguistics, Online, 2020, pp. 1295–1309.

[19] B. He, D. Zhou, J. Xiao, X. Jiang, Q. Liu, N.J. Yuan and T. Xu, BERT-MK: Integrating Graph Contextualized Knowledge into Pre-trained Language Models, in: *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, Online, 2020, pp. 2281–2290. doi:10.18653/v1/2020.findings-emnlp.207. https://aclanthology.org/2020.findings-emnlp.207.

[20] N. Hu, Y. Wu, G. Qi, D. Min, J. Chen, J.Z. Pan and Z. Ali, An empirical study of pre-trained language models in simple knowledge graph question answering, *World Wide Web* **26**(5) (2023), 2855–2886–. doi:10.1007/s11280-023-01166-y.

[21] N. Huang, Y.R. Deshpande, Y. Liu, H. Alberts, K. Cho, C. Vania and I. Calixto, Endowing language models with multimodal knowledge graph representations, *arXiv preprint arXiv:2206.13163* (2022).

[22] X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao, K. Cai, Y. Zhang, S. Wu, P. Xu, D. Wu, A. Freitas and M.A. Mustafa, A survey of safety and trustworthiness of large language models through the lens of verification and validation, *Artificial Intelligence Review* **57**(7) (2024), 175. doi:10.1007/s10462-024-10824-0.

[23] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y.J. Bang, A. Madotto and P. Fung, Survey of Hallucination in Natural Language Generation, *ACM Comput. Surv.* **55**(12) (2023).

[24] M. Johansson and N. Smallbone, Exploring mathematical conjecturing with large language models, *Proceedings of NeSy* (2023).

[25] M. Joshi, D. Chen, Y. Liu, D.S. Weld, L. Zettlemoyer and O. Levy, Spanbert: Improving pre-training by representing and predicting spans, *Transactions of the association for computational linguistics* **8** (2020), 64–77.

[26] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu and M. Huang, JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs, in: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 2526–2538. doi:10.18653/v1/2021.findings-acl.223. https://aclanthology.org/2021.findings-acl.223.

[27] B. Kim, T. Hong, Y. Ko and J. Seo, Multi-Task Learning for Knowledge Graph Completion with Pre-trained Language Models, in: *Proceedings of the 28th International Conference on Computational Linguistics*, D. Scott, N. Bel and C. Zong, eds, International Committee on Computational Linguistics, Barcelona, Spain (Online), 2020, pp. 1737–1743. doi:10.18653/v1/2020.coling-main.153. https://aclanthology.org/2020.coling-main.153.

[28] S. Kukreja, T. Kumar, A. Purohit, A. Dasgupta and D. Guha, A Literature Survey on Open Source Large Language Models, in: *Proceedings of the 2024 7th International Conference on Computers in Management and Business*, Association for Computing Machinery, New York, NY, USA, 2024, pp. 133–143–. ISBN 9798400716652.

[29] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, *Advances in Neural Information Processing Systems* **33** (2020), 9459–9474.

[30] S. Li, X. Li, L. Shang, C. Sun, B. Liu, Z. Ji, X. Jiang and Q. Liu, Pre-training language models with deterministic factual knowledge, *arXiv preprint arXiv:2210.11165* (2022).

[31] S. Li, Y. Gao, H. Jiang, Q. Yin, Z. Li, X. Yan, C. Zhang and B. Yin, Graph reasoning for question answering with triplet retrieval, *arXiv preprint arXiv:2305.18742* (2023).

[32] B.Y. Lin, X. Chen, J. Chen and X. Ren, KagNet: Knowledge-Aware Graph Networks for Commonsense Reasoning, in: *Proceedings of EMNLP-IJCNLP*, K. Inui, J. Jiang, V. Ng and X. Wan, eds, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2829–2839.

[33] Z. Lin, S. Guan, W. Zhang, H. Zhang, Y. Li and H. Zhang, Towards trustworthy LLMs: a review on debiasing and dehallucinating in large language models, *Artificial Intelligence Review* **57**(9) (2024), 243.

[34] Y. Liu, H. He, T. Han, X. Zhang, M. Liu, J. Tian, Y. Zhang, J. Wang, X. Gao, T. Zhong et al., Understanding llms: A comprehensive overview from training to inference, *arXiv:2401.02038* (2024).

[35] D. Lukovnikov, A. Fischer and J. Lehmann, Pretrained Transformers for Simple Question Answering over Knowledge Graphs, in: *The Semantic Web – ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part I*, 2019, pp. 470–486–. ISBN 978-3-030-30792-9.
[36] L. Luo, J. Ju, B. Xiong, Y.-F. Li, G. Haffari and S. Pan, Chatrule: Mining logical rules with large language models for knowledge graph reasoning, *arXiv preprint arXiv:2309.01538* (2023).
[37] X. Lv, Y. Lin, Y. Cao, L. Hou, J. Li, Z. Liu, P. Li and J. Zhou, Do Pre-trained Models Benefit Knowledge Graph Completion? A Reliable Evaluation and a Reasonable Approach, in: *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov and A. Villavicencio, eds, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 3570–3581. doi:10.18653/v1/2022.findings-acl.282. https://aclanthology.org/2022.findings-acl.282.
[38] A. Meyer-Vitali, W. Mulder and M.H. de Boer, Modular design patterns for hybrid actors, *arXiv preprint arXiv:2109.09331* (2021).
[39] B. Min, H. Ross, E. Sulem, A.P.B. Veyseh, T.H. Nguyen, O. Sainz, E. Agirre, I. Heintz and D. Roth, Recent advances in natural language processing via large pre-trained language models: A survey, *ACM Computing Surveys* **56**(2) (2023), 1–40.
[40] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain and J. Gao, Large language models: A survey, *arXiv preprint arXiv:2402.06196* (2024).
[41] T. Mossakowski, Modular design patterns for neural-symbolic integration: refinement and combination, *arXiv:2206.04724* (2022).
[42] N. Muennighoff, T. Wang, L. Sutawika, A. Roberts, S. Biderman, T.L. Scao, M.S. Bari, S. Shen, Z.-X. Yong, H. Schoelkopf et al., Crosslingual generalization through multitask finetuning, *arXiv:2211.01786* (2022).
[43] M. Nayyeri, Z. Wang, M.M. Akter, M.M. Alam, M.R.A.H. Rony, J. Lehmann and S. Staab, Integrating Knowledge Graph Embeddings and Pre-trained Language Models in Hypercomplex Spaces, in: *International Semantic Web Conference*, Springer, 2023, pp. 388–407.
[44] L. Pan, A. Albalak, X. Wang and W.Y. Wang, Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning, *arXiv:2305.12295* (2023).
[45] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang and X. Wu, Unifying large language models and knowledge graphs: A roadmap, *IEEE Transactions on Knowledge and Data Engineering* (2024).
[46] R. Panchendrarajan and A. Zubiaga, Synergizing machine learning & symbolic methods: A survey on hybrid approaches to natural language processing, *Expert Systems with Applications* **251** (2024), 124097.
[47] S. Park and H. Kim, Improving sentence-level relation extraction through curriculum learning, *arXiv preprint arXiv:2107.09332* (2021).
[48] B. Peng, E. Alcaide, Q. Anthony, A. Albalak, S. Arcadinho, S. Biderman, H. Cao, X. Cheng, M. Chung, M. Grella et al., Rwkv: Reinventing rnns for the transformer era, *arXiv preprint arXiv:2305.13048* (2023).
[49] M.E. Peters, M. Neumann, R.L. Logan IV, R. Schwartz, V. Joshi, S. Singh and N.A. Smith, Knowledge enhanced contextual word representations, *arXiv:1909.04164* (2019).
[50] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever et al., Language models are unsupervised multitask learners, *OpenAI blog* **1**(8) (2019), 9.
[51] C. Rosset, C. Xiong, M. Phan, X. Song, P. Bennett and S. Tiwary, Knowledge-aware language model pretraining, *arXiv preprint arXiv:2007.00655* (2020).
[52] G. Rossiello, M.F.M. Chowdhury, N. Mihindukulasooriya, O. Cornec and A.M. Gliozzo, KnowGL: Knowledge Generation and Linking from Text, in: *AAAI*, 2023, pp. 16476–16478.
[53] T. Shen, Y. Mao, P. He, G. Long, A. Trischler and W. Chen, Exploiting structured knowledge in text via graph-guided representation learning, *arXiv preprint arXiv:2004.14224* (2020).
[54] P. Shi and J. Lin, Simple bert models for relation extraction and semantic role labeling, *arXiv preprint arXiv:1904.05255* (2019).
[55] Y. Su, X. Han, Z. Zhang, Y. Lin, P. Li, Z. Liu, J. Zhou and M. Sun, CokeBERT: Contextual knowledge selection and embedding towards enhanced pre-trained language models, *AI Open* **2** (2021), 127–134.
[56] J. Sun, C. Xu, L. Tang, S. Wang, C. Lin, Y. Gong, H.-Y. Shum and J. Guo, Think-on-Graph: Deep and Responsible Reasoning of Large Language Model on Knowledge Graph, 2023.
[57] Y. Sun, Q. Shi, L. Qi and Y. Zhang, JointLK: Joint Reasoning with Language Models and Knowledge Graphs for Commonsense Question Answering, 2022, pp. 5049–5060.
[58] H. Tian, C. Gao, X. Xiao, H. Liu, B. He, H. Wu, H. Wang and F. Wu, SKEP: Sentiment knowledge enhanced pre-training for sentiment analysis, *arXiv preprint arXiv:2005.05635* (2020).
[59] M. van Bekkum, M. de Boer, F. van Harmelen, A. Meyer-Vitali and A.t. Teije, Modular design patterns for hybrid learning and reasoning systems: a taxonomy, patterns and use cases, *Applied Intelligence* **51**(9) (2021), 6528–6546.
[60] F. Van Harmelen and A. Ten Teije, A boxology of design patterns for hybrid learning and reasoning systems, *Journal of Web Engineering* **18**(1–3) (2019), 97–123.
[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser and I. Polosukhin, Attention is all you need, *Advances in neural information processing systems* **30** (2017).
[62] J. Wang, Q. Sun, X. Li and M. Gao, Boosting language models reasoning with chain-of-knowledge prompting, *arXiv preprint arXiv:2306.06427* (2023).
[63] P. Wang, X. Xie, X. Wang and N. Zhang, Reasoning through memorization: Nearest neighbor knowledge graph embeddings, in: *CCF International Conference on Natural Language Processing and Chinese Computing*, Springer, 2023, pp. 111–122.
[64] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li and J. Tang, KEPLER: A unified model for knowledge embedding and pre-trained language representation, *Transactions of the Association for Computational Linguistics* **9** (2021), 176–194.
[65] X. Wang, Q. He, J. Liang and Y. Xiao, Language models as knowledge embeddings, *arXiv preprint arXiv:2206.12617* (2022).

[66] Y. Wang, N. Lipka, R. Rossi, A. Siu, R. Zhang and T. Derr, Knowledge Graph Prompting for Multi-Document Question Answering, *Proceedings of the AAAI Conference on Artificial Intelligence* **38** (2024), 19206–19214.

[67] X. Wei, S. Wang, D. Zhang, P. Bhatia and A. Arnold, Knowledge enhanced pretrained language models: A compreshensive survey, *arXiv:2110.08455* (2021).

[68] Y. Wen, Z. Wang and J. Sun, Mindmap: Knowledge graph prompting sparks graph of thoughts in large language models, *arXiv preprint arXiv:2308.09729* (2023).

[69] S. Williams and J. Huckle, Easy Problems That LLMs Get Wrong, *arXiv preprint arXiv:2405.19616* (2024).

[70] T. Wu, S. He, J. Liu, S. Sun, K. Liu, Q.-L. Han and Y. Tang, A Brief Overview of ChatGPT: The History, Status Quo and Potential Future Development, Vol. 10, 2023, p. 1122. ISSN 2329-9266.

[71] X. Xie, Z. Li, X. Wang, Z. Xi and N. Zhang, Lambdakg: A library for pre-trained language model-based knowledge graph embeddings, *arXiv preprint arXiv:2210.00305* (2022).

[72] W. Xiong, J. Du, W.Y. Wang and V. Stoyanov, Pretrained encyclopedia: Weakly supervised knowledge-pretrained language model, *arXiv preprint arXiv:1912.09637* (2019).

[73] H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang and X. Qiu, A Unified Generative Framework for Various NER Subtasks, in: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 5808–5822. doi:10.18653/v1/2021.acl-long.451. https://aclanthology.org/2021.acl-long.451.

[74] S. Yang, X. Li, L. Cui, L. Bing and W. Lam, Neuro-Symbolic Integration Brings Causal and Reliable Reasoning Proofs, 2023.

[75] L. Yao, C. Mao and Y. Luo, KG-BERT: BERT for knowledge graph completion, *arXiv preprint arXiv:1909.03193* (2019).

[76] M. Yasunaga, H. Ren, A. Bosselut, P. Liang and J. Leskovec, QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering, in: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, 2021, pp. 535–546.

[77] M. Yasunaga, A. Bosselut, H. Ren, X. Zhang, C.D. Manning, P.S. Liang and J. Leskovec, Deep bidirectional language-knowledge graph pretraining, *Advances in Neural Information Processing Systems* **35** (2022), 37309–37323.

[78] D. Yu, C. Zhu, Y. Yang and M. Zeng, Jaket: Joint pre-training of knowledge graph and language understanding, in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36, 2022, pp. 11630–11638.

[79] A. Zeng, M. Liu, R. Lu, B. Wang, X. Liu, Y. Dong and J. Tang, Agenttuning: Enabling generalized agent abilities for llms, *arXiv preprint arXiv:2310.12823* (2023).

[80] D. Zhang, Z. Yuan, Y. Liu, F. Zhuang, H. Chen and H. Xiong, E-BERT: A phrase and product knowledge enhanced language model for e-commerce, *arXiv preprint arXiv:2009.02835* (2020).

[81] X. Zhang, A. Bosselut, M. Yasunaga, H. Ren, P. Liang, C.D. Manning and J. Leskovec, GreaseLM: Graph reasoning enhanced language models for question answering, *arXiv preprint arXiv:2201.08860* (2022).

[82] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun and Q. Liu, ERNIE: Enhanced language representation with informative entities, *arXiv preprint arXiv:1905.07129* (2019).

[83] Z. Zhang, X. Liu, Y. Zhang, Q. Su, X. Sun and B. He, Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models, in: *Findings of the Association for Computational Linguistics: EMNLP 2020*, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, Online, 2020, pp. 259–266. doi:10.18653/v1/2020.findings-emnlp.25. https://aclanthology.org/2020.findings-emnlp.25.

[84] H. Zhao, H. Chen, F. Yang, N. Liu, H. Deng, H. Cai, S. Wang, D. Yin and M. Du, Explainability for large language models: A survey, *ACM Transactions on Intelligent Systems and Technology* **15**(2) (2024), 1–38.