Knowledge Graphs and Knowledge Infusion in Language Models

Tanu Goyal, Pushpak Bhattacharyya
Department of Computer Science and Engineering, IIT Bombay
{tanugoyal, pb}@cse.iitb.ac.in

Abstract

With the explosion of digital data, there is an increasing need to organize, structure, and make sense of the vast amount of information available. One way to achieve this is through knowledge graphs, which represent knowledge as a network of interconnected nodes and edges. Language models, on the other hand, are natural language processing systems that learn to generate human-like text by training on large amounts of data. In recent years, there has been a growing interest in combining these two technologies to create knowledge-infused language models that can reason and generate text based on structured knowledge. This paper explores the concepts knowledge infusion in language models, with a focus on the role of knowledge graphs in enhancing the capabilities of LMs. It reviews recent research on knowledge graph integration in LMs, discussing the benefits and challenges of this approach, and highlighting the potential for future developments. Overall, the paper provides insights into the use of knowledge graphs as a powerful tool for enhancing the capabilities of LMs and highlights the potential impact of this approach on the field of natural language processing.

1 Introduction

The integration of knowledge graphs and LMs has the potential to transform natural language processing by enabling models to reason and generate text based on structured knowledge. By incorporating knowledge graphs into LMs, models can access a vast repository of structured knowledge, allowing them to generate more accurate and coherent responses. Additionally, knowledge-infused LMs can provide explanations for their output, enabling greater transparency and interpretability.

1.1 Problem Statement

Despite the potential benefits of integrating knowledge graphs in LMs, there are still many challenges that need to be addressed. For instance, the integration of knowledge graphs requires significant computational resources and can be time-consuming. Additionally, the accuracy and relevance of the knowledge provided by knowledge graphs need to be carefully evaluated to avoid negatively impacting the performance of LMs. Therefore, this paper aims to review recent research on knowledge graph integration in LMs, discuss the benefits and challenges of this approach, and highlight the potential for future developments in this field.

1.2 Motivation

Language models have made significant progress in recent years, but they are still limited in their ability to reason about complex relationships between entities and concepts. The use of knowledge graphs as a structured way of representing knowledge offers a promising approach to address this limitation. Knowledge graphs can provide LMs with external knowledge that can be used to reason and generate text. The integration of knowledge graphs in LMs has the potential to significantly enhance their capabilities and make them more accurate and effective in generating text. Knowledge Graphs have domain-specific as well as general information. While language models hallucinate etc., they can deal with natural language queries. To answer queries using a KG, the query needs to follow the format of the KG using particular/specific relation names, etc., but using an LM with that KG infused can help eliminate this issue. Further having a domain specific KG is useful in terms of reducing redundancy, training time and inference latency. Further, it's unclear from the literature how KG infusion impacts different domains in different training settings.

2 Knowledge Graphs

In knowledge representation and reasoning, knowledge graph is a knowledge base that uses a graph-structured data model or topology to integrate data. Knowledge graphs are often used to store interlinked descriptions of entities – objects, events, situations or abstract concepts – while also encoding the semantics underlying the used terminology. Knowledge graphs play an important role in natural language processing (NLP) by providing a structured representation of knowledge that can be used to enhance understanding and interpretation of natural language text. Knowledge graphs are used to build semantic models that capture the relationships between words and concepts in a text. This is done by mapping the entities and relation- ships mentioned in a text to the nodes and edges in a knowledge graph. For example, if we have a sentence like *"John works at Google,"* a knowledge graph can represent *John* and *Google* as nodes, with an edge between them representing the *"works at"* relationship. This representation can be used to answer questions like *"Where does John work?"* or *"Who works at Google?"* Knowledge graphs can also be used to disambiguate ambiguous terms in natural language text. For example, the term *"apple"* can refer to a fruit or a technology company. By using a knowledge graph, it is possible to determine which meaning of *"apple"* is most relevant in a particular context. One of the key advantages of knowledge graphs in NLP is that they can be used to connect concepts across different domains and datasets. This allows for more comprehensive understanding of complex topics that span multiple areas of knowledge. Overall, knowledge graphs are a powerful tool for enhancing the capabilities of natural language processing systems, and are becoming increasingly important in the development of advanced NLP applications.

In the following sections, we discuss some of the major knowledge graphs including ConceptNet and WordNet.

2.1 ConceptNet

ConceptNet 5.5 (Speer et al., 2017) presented a new version of the linked open data resource ConceptNet. ConceptNet is a multilingual knowledge graph that captures semantic relations between natural language words and phrases through labeled edges. The knowledge captured in ConceptNet includes both expert-created resources as well as crowd-sourced resources. ConceptNet primarily captures the syntagmatic relations between words.ConceptNet connects words and phrases of natural language (referred to as terms) with labeled, weighted edges (called assertions). ConceptNet incorporates knowledge from a variety of sources, including Open Mind Common Sense, Wiktionary, Open Multilingual WordNet, etc. with Wikitonary being the largest source of input. ConceptNet contains more than 8 million nodes and over 21 million edges combined. Its English vocabulary has around 1,500,000 nodes, and there are 83 languages in which it contains at least 10,000 nodes. ConceptNet supports 36 relations and the edges representing these relations are directed in nature. The relations are further classified into two types, symmetric and asymmetric relations. For the symmetric relations, direction of edges is irrelevant. Using ConceptNet, a semantic space was build that is more effective than distributional semantics alone. Number-batch embeddings, constructued using the retrofitting technique, capture the ConceptNet graph structure as well as the distributional semantics as captured by GloVE, word2vec, etc.

2.2 WordNet

WordNet (Miller, 1994) is a lexical database for the English language that was developed at Princeton University in the 1980s. It is a large, electronic dictionary that groups words into sets of synonyms, called synsets, and provides short definitions for each of them. WordNet has become a valuable resource for natural language processing and computational linguistics, as it allows computers to understand the relationships between words and to make inferences based on that knowledge.

The database is organized into four main components (based on the lexical category): nouns, verbs, adjectives, and adverbs. Each of these components is further divided into synsets, or sets of synonyms that share the same meaning. For example, the noun component includes synsets for *"cat," "feline," "kitten,"* and *"tomcat,"* among others. Each synset is given a unique identifier, called a synset ID, and is associated with a short definition, referred to as gloss, that captures the core meaning of the set. The database contains 155,327 words organized in 175,979 synsets.

WordNet also includes a rich set of relationships between synsets. These relationships, being paradigmatic in nature, can be used to make inferences about the meanings of words and to perform tasks such as word sense disambiguation, which is the process of determining which meaning of a word is intended in a particular context. Some of the key relationships in WordNet include hypernymy (the relationship between a more general word and a more specific word, e.g., "*cat*" is a hypernym of "*siamese*"), hyponymy (the inverse of hypernymy), meronymy (the relationship between a whole and its parts, e.g., "*car*" is a meronym of "*wheel*"), and holonymy (the inverse of meronymy).

WordNet has been used in a wide range of applications, from information retrieval to machine translation to sentiment analysis. It has also been extended to other languages, with databases now available for languages such as Spanish, German, and Italian. While WordNet has its limitations, particularly in terms of its coverage and accuracy, it remains a valuable resource for researchers and developers working in natural language processing and related fields.

2.3 Wikidata

Wikidata (Vrandečić and Krötzsch, 2014) is a vast and growing knowledge graph that contains structured data on a wide range of topics. It is an open, collaborative, and multilingual database that can be accessed and edited by anyone. With its rich and diverse data, Wikidata is a valuable resource for many natural language processing (NLP) tasks.

The utility of Wikidata for NLP tasks lies in its ability to represent complex relationships between entities. For example, an item can have multiple types of relationships with other items, such as *"instance of", "subclass of", "part of",* and *"has part".* This flexibility allows for the creation of rich and comprehensive knowledge graphs. Entities in Wikidata are represented as items, and each item can have a variety of properties that describe its attributes and relationships with other items. This structured data makes it easier for NLP systems to extract and understand information from text.

2.4 FoodKG

FoodKG (Haussmann et al., 2019) is a knowledge graph that aims to represent and link information about food and nutrition. It is a comprehensive database that integrates data from various sources, including scientific publications, nutritional databases, food labels, and user-generated content, to provide a more holistic understanding of the food domain.

The FoodKG database is structured as a graph, where nodes represent food items, nutrients, or other relevant entities, and edges represent relationships between them. This graph-based approach allows for more flexible and efficient querying and analysis of the data. By leveraging the rich information in the graph, FoodKG can generate tailored meal plans, suggest substitutions for allergens or dietary restrictions, and provide nutritional information for various food items.

Another important application of FoodKG is in food safety and quality control. By linking information about food items, production processes, and supply chains, FoodKG can help identify potential sources of contamination or quality issues, and track them to their origin.

3 Language Modeling

Language modeling is a fundamental task in NLP that involves predicting the likelihood of a sequence of words in a given language. Models that assign probabilities to sequences of words are called Language Models(LMs). For example, there are multiple possible continuations for the phrase *I cleaned the table* ___ . Words like *yesterday, twice, today* form a more probable continuation while words like *okay, table, your* lead to less probable continuations. Language modeling captures the notion of varied probabilities for different words i.e.

$$P(w_n|w_{n-1}...w_1)$$

Given a sequence of words, language models provide a probability distribution for the next word. LMs can also model sentence probabilities. Say we have a sentence of the form $w_1w_2...w_n$ then

$$P(w_1...w_n) = P(w_n|w_{n-1}...w_1).... * P(w_1)$$

3.1 N-gram Language Modeling

n-gram refers to a contiguous sequence of *n* words. n-gram based LMs provide a probability distribution for the next word using previous n-1 words. Given a sequence of k words $w_1....w_k$,

$$Pr(w_{k+1}|w_k...w_1) \approx Pr(w_{k+1}|w_k...w_{k-n-1})$$

If n=1 (Unigram language modeling), then the next word depends only on the immediately preceding word. This is also referred to as the Markov Assumption.

$$Pr(w_{k+1}|w_k...w_1) \approx Pr(w_{k+1}|w_k)$$

If n=k, then the next word depends on all the previous words. This is often referred to as an Autoregressive Model.

3.2 RNN/LSTMs

As the length of the input grows, the performance of N-gram model decreases. We need a model that can deal with any length input. RNNs have the ability to handle varying length input. Also, RNNs are suitable for sequential data. Hence, RNNs became an obvious choice for language modeling. Like any other neural model, RNNs are trained on large corpus by optimizing cross-entropy loss. Figure 1 shows different types of RNNS. LSTMs are special type of RNNs that are enable learning long-term dependencies.

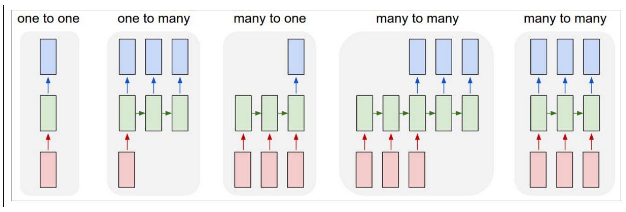

Figure 1: Types of RNNs

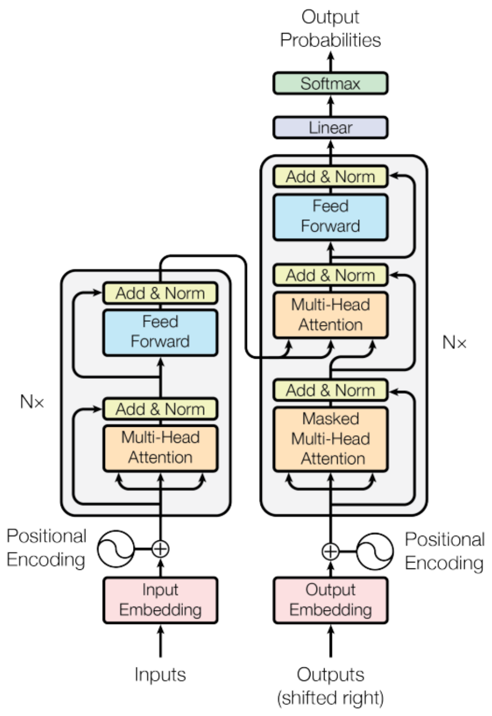

Figure 2: Transformer - model architecture (Vaswani et al., 2017)

3.3 Transformer

(Vaswani et al., 2017) proposed transformer architecture, which is based solely on attention mechanism. Transformers eliminated the need for recurrences for creating powerful language models. Figure 2 shows the transformer architecture. It is an encoder-decoder architecture, where the decoder is autoregressive. This was based on a novel concepts of multi head attention, whereby model can jointly attend to information from different representations at different positions (using positional encodings).

Transformers offer multiple advantages over the existing attention based architectures. It makes input encoding parallelizable, thus reducing the training time significantly. Furthermore, the number of operations required to connect an input to any output is constant – unlike RNN/LSTMs where it grows linearly with the distance between input and output.

4 Large Language Models

Large Language Models(LLMs) are language models that are trained on enormous amount of data by using humongous amount of computation power. By virtue of the large amount of data spread across various domains and billions of parameters, LLMs capture syntactic structure and are able to learn some factual knowledge too. This enables LLMs to perform a variety of tasks without separately fine tuning for each task, or zero-shot learning as it is commonly referred to as. Large Language Models (LLMs) are language models that have seen huge amount of data while pre-training. LLMs are, in general, pre-trained in a self-supervised manner, whereby the target value can be computed given the input and the objective function, thus dispensing the need for labeling the humongous pre-training data. Based on the underlying model architecture, LLMs can be categorized into three classes - *auto-encoding*, *auto regressive* and *sequence-to-sequence*. In this chapter, we discuss the three classes of LLMs and the leading LLMs in each class.

4.1 Encoder-only models

Encoder-only models are the language models that consist of only the encoder component of transformer architecture, discarding the decoder component. In order to pre-train encoder-only models, the input sentence is corrupted by adding noise to it, and the model's goal is to reconstruct the original (uncorrupted) sentence. While reconstructing any word, the model has access to the context words in both the directions. The most common ways of corrupting input text are masking words or phrases, or re-ordering parts of sentences. By virtue of the pre-training process, these models learn language properties. As the network is trained to ignore signal noise, these models also learn efficient data representations of sentences (while retaining its original sense) and are therefore often referred to as auto-encoding models. Encoder-only models are best suited for tasks that require a grasp of the entire sentence, like sentence classification, extractive question answering and named entity recognition. Some representatives of this family of models are:

- BERT: Bidirectional Encoder Representations from Transformers
- ALBERT : A Lite BERT
- DistilBERT: Distilled version of BERT
- ELECTRA: Efficiently Learning an Encoder that Classifies Token Replacements Accurately
- RoBERTa : Robustly Optimized BERT Pretraining Approach

BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2018)is one of the earliest LLMs. BERT is first pre-trained using two objectives and then fine-tuned for a specific task. Figure 3 shows the pre-training and fine-tuning procedures for BERT.
BERT uses two strategies for training:

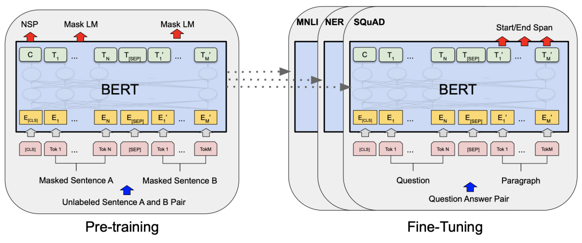

Figure 3: BERT pre-training and fine-tuning (Devlin et al., 2018)

- **Masked LM (MLM)**: Word sequences are changed with a [MASK] token for 15% of the words in each sequence before being fed into the BERT. Based on the context given by the other, non-masked, words in the sequence, the model then makes an attempt to predict the original value of the masked words. The prediction of the non-masked words is disregarded by the BERT loss function, which only considers the prediction of the masked values.
- **Next Sentence Prediction (NSP)**: In the BERT training phase, the model learns to predict whether the second sentence in a pair will come after another in the original document by receiving pairs of sentences as input. During training, 50% of the inputs are pairs in which the second sentence is the next one in the original text, and in the remaining 50%, the second sentence is a randomly selected sentence from the corpus. The underlying presumption is that the second phrase will not be connected to the first.

BERT can be fine-tuned for the downstream task by simply adding a single layer on top of the core model, we may adjust the original model depending on our own dataset. Figure 4 shows fine tuning BERT on various downstream tasks.

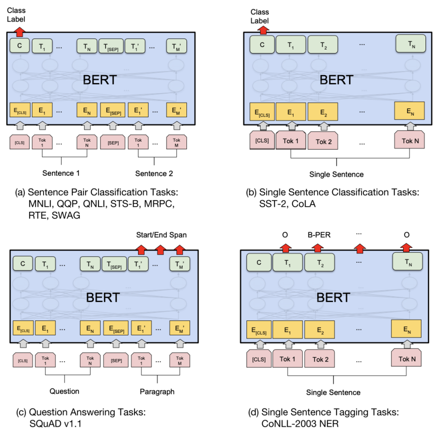
Class Label
BERT
Sentence 1
Sentence 2
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG
Class Label
BERT
Single Sentence
(b) Single Sentence Classification Tasks: SST-2, CoLA
Start/End Span
BERT
Question
Paragraph
(c) Question Answering Tasks: SQuAD v1.1
O
B-PER
O
BERT
Single Sentence
(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Figure 4: Fine tuning BERT on various downstream tasks (Devlin et al., 2018)

4.2 Decoder-only models

Decoder-only models are the language models that consist of only the decoder component of transformer architecture, discarding the encoder component. Only the words that come before a certain word in the sentence can be accessed by the attention layers at each step. These are frequently referred to as auto-regressive models. The typical focus of pretraining for decoder models is on predicting the next word that will be used. These models work best for text generation tasks. GPT-2 (Radford et al., 2019), GPT-3 and BLOOM are some of the most successful auto regressive models.
GPT-2 is a decoder only model that at the time of its release stood out for having 1.5 billion trainable parameters. The model is pretrained using

text from 45 million webpage links in the WebText dataset. With certain changes, it essentially adheres to the prior GPT architecture. The following are the two crucial ideas that it alluded to:

- **Task Conditioning:** As we previously observed, the language model's training objective is written as $P(output|input)$. GPT-2, on the other hand, sought to learn many tasks using the same unsupervised model. The learning objective should be changed to $P(output|input, task)$ to accomplish that. The model is expected to give different output for the same input for different tasks, and this adjustment is referred to as task conditioning. Some models incorporate task conditioning at the architectural level, feeding the model the task as well as the input. For language models, the job, input, and output are all in natural language. As a result, task conditioning for language models is carried out by giving the model examples or instructions in natural language.

- **Zero Shot Learning and Zero Short Task Transfer:** GPT 2's capacity to transfer zero shot tasks is intriguing. As a special case of zero shot task transfer, zero shot learning occurs when no examples are given at all and the model is instructed to perform the task. The format of the input to GPT-2 assumed that the model would comprehend the nature of the assignment and suggest solutions. To mimic zero-shot task transfer behaviour, this was done. For instance, the model was given an English sentence, followed by the word French, and a prompt for the English to French translation assignment (:). The model was expected to comprehend that the task involved translation and provide the French equivalent of the English statement.

GPT-2 has 1.5 billion parameters, 48 layers and used 1600 dimensional vectors for word embedding. Layer normalisation is done to input of each sub-block and an additional layer normalisation is added after final self-attention block.

5 Encoder-decoder models

The Transformer architecture is used by encoder-decoder models, often known as sequence-to-sequence models. The attention layers of the encoder can access every word in the first sentence at every step, whereas the attention layers of the decoder can only access the input words that come before a certain word.

It is possible to pretrain these models using the goals of encoder or decoder models, although this typically entails something more complicated. For instance, T5 is pre-trained by substituting a single mask special word for random text segments (which may contain numerous words), with the goal being to anticipate the text that this mask word substitutes. The tasks that revolve around creating new sentences from an input, such summarization, translation, or generative question answering, are best suited for sequence-to-sequence models.

Some of the LLMs that follow encoder-decoder architecture are BART, mBART, T5, to name a few. The T5 model was presented in the paper (Raffel et al., 2020) . Figure 5 depicts the T5 framework.

Every task taken into consideration—including

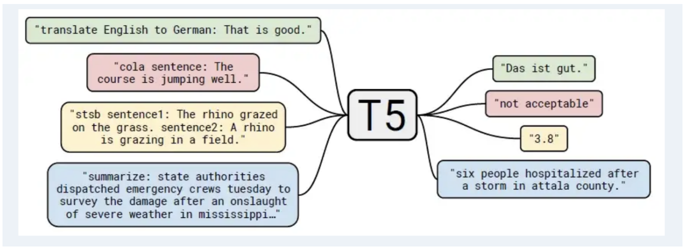
"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 5: T5 framework(Raffel et al., 2020)

translation, question-answering, and classification—is framed as feeding the T5 model text as input and training it to output some destination text. T5 stands for "Text-to-Text Transfer Transformer." Figure 6 illustrates the architecture of T5.

With the exception of the following modifications,

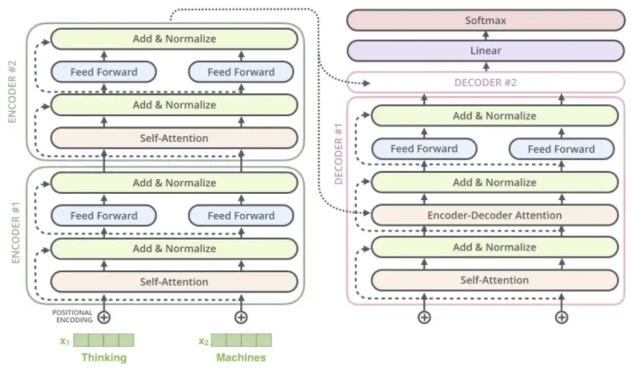

Figure 6: Illustration of T5 architecture(Raffel et al., 2020)

T5 uses an encoder-decoder Transformer implementation that substantially resembles the original

Transformer:

- A residual skip connection, originating from ResNet, adds each subcomponent's input to its output after layer normalisation. With only a rescaling of the activations and no additive bias, a streamlined layer normalisation is used.

- Additionally, relative position embeddings provide a distinct learned embedding depending on the offset (distance) between the "key" and "query" being compared in the self-attention process, as opposed to utilising a fixed embedding for each position.

On numerous benchmarks encompassing summarization, question-answering, text classification, and more, T5 achieves state-of-the-art results.

6 Knowledege Infusion into Language Models

LLMs and Knowledge Graphs can be used together to improve various NLP tasks such as question answering, information retrieval, and text generation. A Knowledge Graph captures the relationships between entities and concepts in a domain. On the other hand, LLMs can generate language by predicting the next word in a sequence of text, and they are trained on large datasets of text to learn patterns and structures of language.

The fusion of LLMs and KGs can improve the performance of NLP systems. For example, by integrating KGs into LLMs, it is possible to improve their ability to answer complex questions that require background knowledge. The LLMs can use the KGs to identify relevant entities and relationships between them to generate more accurate and relevant responses.

Similarly, KGs can be used to augment LLMs' training data, leading to better language models. The KGs provide a structured representation of domain-specific knowledge, which can be used to generate more coherent and consistent text. Moreover, LLMs can also be used to enrich KGs by extracting and linking entities from unstructured text. This process can help in populating the KGs with additional information, improving their completeness and accuracy. In the following sub-sections we review some of the recent works focused on augmenting LLMs with KGs.

6.1 QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering

(Yasunaga et al., 2021) proposed a novel end-to-end question answering model called QA-GNN that utilizes both language models (LM) and knowledge graphs (KG). Their model includes two key innovations. Firstly,an LM is used to encode the QA context and retrieve a KG subgraph following previous research. A relevance scoring mechanism is employed to compute the relevance of KG nodes based on the given context of the question answering (QA) task. Relevance scoring scores each entity on the KG subgraph by concatenating it with the QA context and using a pre-trained LM to calculate the likelihood. This allows us to weight the information on the KG based on its relevance to the given context; and Joint reasoning, where a joint graph representation of the QA context and KG is created by treating the QA context as an additional node and connecting it to the topic entities in the KG subgraph. This unified graph, referred to as the working graph, enables us to update the representation of both the KG entities and the QA context node simultaneously using a new attention-based GNN module, bridging the gap between the two sources of information. (Yasunaga et al., 2021) have shown through both quantitative and qualitative analyses that QA-GNN outperforms existing LM and LM+KG models on question answering tasks, and that it is capable of performing interpretable and structured reasoning, such as correctly handling negation in questions.

6.2 GreaseLM: Graph Reasoning Enhanced Language Models for Question Answering

In order to answer complex questions about textual narratives, it is necessary to reason over both the contextual information presented in the text and the underlying world knowledge. However, current pre-trained language models (LM) used in most modern question answering (QA) systems do not adequately represent the latent relationships between concepts required for effective reasoning. While knowledge graphs (KG) can be used to supplement LM with structured representations of world knowledge, the challenge lies in effectively combining and reasoning over both the KG representations and language context. (Zhang et al., 2022) purposed a new model called GreaseLM that utilizes multiple layers of modality interaction

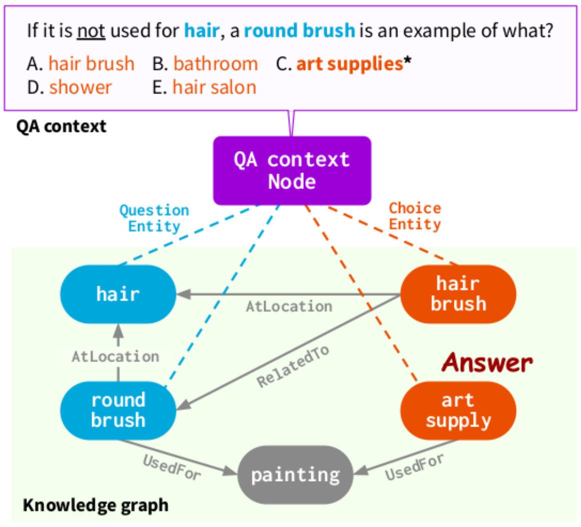

Figure 7: Nodes corresponding to question and answer concepts are connected to a qa-context node which contains the information from the LM encoder(Yasunaga et al., 2021)

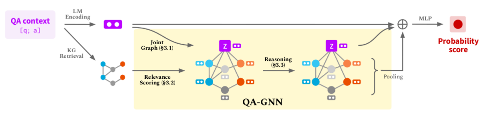

Figure 8: QAGNN architecture: Retrieved KG is connected to the QA context and relevance of KG nodes is computed conditioned on the QA context. Post per-node relevance scoring, reasoning is done over the entire graph. (Yasunaga et al., 2021)

operations to fuse encoded representations from pretrained LMs and graph neural networks. This allows information to flow between the modalities, enabling the grounding of language context representations with structured world knowledge and allowing linguistic nuances in the context to inform the graph representations of knowledge. Their results on benchmark datasets in the domains of commonsense reasoning and medical question answering demonstrate that GreaseLM can answer questions that require reasoning over both situational constraints and structured knowledge with greater reliability, even outperforming models that are 8 times larger.

6.2.1 Architecture

The GreaseLM model is composed of both a language model (LM) that processes the natural language context and a graph neural network (GNN) that aids in performing reasoning over the knowledge graph (KG). At each layer of the LM and

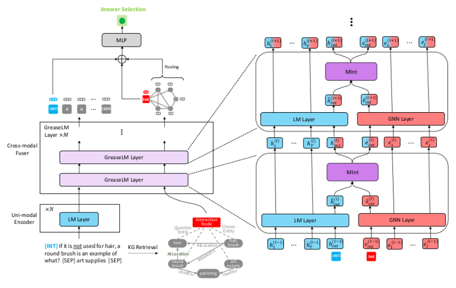

Figure 9: GreaseLM architecture (Zhang et al., 2022)

GNN, there is a bi-directional transfer of information between the two modalities facilitated by specially initialized interaction representations. This transfer of information occurs through the interaction token, which allows all tokens in the language context to receive information from the KG entities, and the interaction node, which enables the KG entities to indirectly interact with the tokens.

Performance improvements have been reported (Zhang et al., 2022) using the GreaseLM architecture on various MCQA benchmarks. Given an instance of the form (question, list of candidate answers, ground truth answer)from a MCQA benchmark, a confidence score for each of the candidate answers is calculated separately and the answer with the highest confidence score is output as the prediction.

The question and the candidate answer are concatenated together and fed as input to the GreaseLM architecture. In case question context is available, it is appended to the input as well. The key concepts are first extracted from the input and these are further used to extract a very small subgraph from a large knowledge graph like ConceptNet. The intuition here is to extract and process only part of the KG which is relevant to the query or the candidate answers. Once we have the input text and the corresponding KG subgraph, we obtain the initial embeddings for the two modalities namely text and the KG are obtained using a transformer based LM and a GNN respectively. These embeddings are then fed to the cross-modal GreaseLM layers where special token and nodes are used to ensure information fusion between the two modalities. The fused representations of text and KG are then used to assign a score to the input candidate answer.

The paper proposed a novel model called GreaseLM that facilitates interactive fusion by enabling joint information exchange between lan-

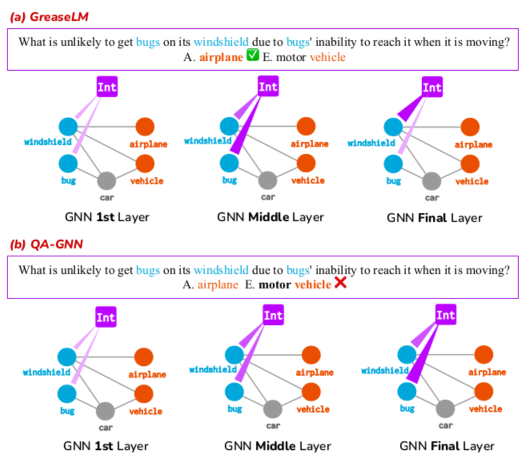
(a) GreaseLM
What is unlikely to get bugs on its windshield due to bugs' inability to reach it when it is moving?
A. airplane ✅ E. motor vehicle
Int
windshield
airplane
bug
car
vehicle
GNN 1st Layer
GNN Middle Layer
GNN Final Layer
(b) QA-GNN
What is unlikely to get bugs on its windshield due to bugs' inability to reach it when it is moving?
A. airplane E. motor vehicle ✗
GNN 1st Layer
GNN Middle Layer
GNN Final Layer

Figure 10: Qualitative analysis of GreaseLM (Zhang et al., 2022)

guage models and knowledge graphs. The model outperforms existing architectures on multiple benchmarks and also performs better in modeling questions that exhibit textual nuances, such as negation and hedging.

6.3 SKILL: Structured Knowledge Infusion for Large Language Models

SKILL (Moiseev et al., 2022) explores knowledge-augmented LM pre-training. Instead of providing access to an external KG, SKILL infuses the knowledge from these KGs into the LM itself. SKILL pre-trains language models using masked (subject, relation, object) triples of the KG of interest. In the following subsections, we discuss the techniques and experiments presented in the SKILL paper.

6.3.1 Method

The triples of a knowledge graph of the form (subject, relation, object) are converted into a sentence by concatenating the subject, relation, object together with space as delimiter. For each generated sentence either the subject or the object is masked, with an equal probability of 0.5 each. Next, sentences are taken from a natural language corpora like C4 and salient making is applied to sentences from these corpora. The masked sentences from both C4 and knowledge triples are mixed in a 50:50 ratio and used for training a pre-trained language model like T5-large. The so-trained model can be fine-tuned for specific downstream tasks. The technique suggested by (Moiseev et al., 2022) incorporates knowledge from knowledge graphs into language models by training the pre-trained language models on knowledge graph triples and also empirically establishes that models that have been pre-trained on factual triples perform on par with models that have been trained on phrases in natural language that contain the same information. Since KG triples needn't be verbalised and can be used directly by concatenating the subject, relation and the object, this technique is easily applicable to industry-scale KGs.

6.3.2 Limitations

Some of the limitations of this work include: This technique of infusing knowledge into LMs only adds the capability to answer factoid questions i.e. the questions for which the answer forms part of the KG triple. Further, the performance improvements for multi-hop questions is not so significant, since the knowledge infusion step involves training using single triples, and the model may fail to reason and capture multi-hop relation among triples.

6.4 Knowledge Infused Decoding

The existing pre-trained language models (LMs) can memorize a significant amount of knowledge from their training data, but they are still limited in recalling factually correct information in a given context. This limitation causes them to produce counterfactual or hallucinatory output for knowledge-intensive natural language generation (NLG) tasks. Current solutions to this problem include additional pre-training, fine-tuning, or architecture modification of LMs to incorporate external knowledge.
To address this problem, (Liu et al., 2022) propose a new decoding algorithm called Knowledge Infused Decoding (KID) for generative LMs. KID dynamically incorporates external knowledge into each step of the LM decoding process by maintaining a local knowledge memory based on the current context, which dynamically interacts with a created external knowledge trie, continuously updating the local memory. The local memory poses knowledge-aware constraints to guide decoding in a reinforcement learning environment.
(Liu et al., 2022) demonstrate KID's effectiveness on six diverse knowledge-intensive NLG tasks, where task-agnostic LMs equipped with KID (such as GPT-2 and BART) outperform many task-specific state-of-the-art models. KID also shows particularly strong performance in few-shot scenarios over seven related knowledge-infusion techniques. Human evaluations confirm KID's improved capability for eliminating hallucinations

compared to multiple baselines. Finally, KID addresses exposure bias and generates stable output quality when producing longer sequences.

6.4.1 Architecture

KID's architecture consists of three modules, as shown in Figure 11:

- Knowledge Retrieval: This module retrieves top-k relevant documents for a given context query; which are subsequently used to ground the text generation. Dense passage based Retriever, a BERT based bi-encoder, is used for document retrieval. Maximum inner-product search is used in the retrieval process.

- Memory Construction: The retrieval step retrieves multiple documents for a given context vector. Memory Construction module processes the retrieved documents to ease identification of relevant knowledge and reduces the memory foot print by compressing the knowledge using specialized data structures. The documents are converted into knowledge triples and construct a trie on top of the extracted knowledge triples, to ease knowledge search.

- Interaction Guided Decoding: Existing LMs are trained to maximize the probability of current token at each decoding step, but the training objectives don't constrain the output to be factually grounded. Thus the generated text is prone to hallucinations. To address this the decoding step is formulated as an Reinforcement Learning problem, where higher rewards are incurred on factually correct generations. The knowledge trie is queried using the local memory, and the rewards are assigned based on the retrieved results.

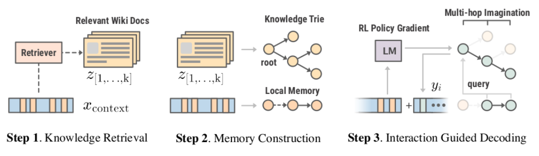

Figure 11: Overview of KID decoding algorithm (Liu et al., 2022)

The KID technique a knowledge guided decoding step with zero architectural changes to existing models. The memory construction step may be made more efficient using other data structures, and different reward types for the decoding step may be explored.

7 Summary

This paper reviews recent research on knowledge graph integration in LMs, covering a range of approaches, from simple pre-training on knowledge graphs to more complex methods that incorporate knowledge graph embeddings directly into the LM. The paper discusses the benefits and challenges of the current apporaches and highlights potential avenues for future research.

References

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*.

Steven Haussmann, Oshani Seneviratne, Yu Chen, Yarden Ne'eman, James Codella, Ching-Hua Chen, Deborah L McGuinness, and Mohammed J Zaki. 2019. Foodkg: a semantics-driven knowledge graph for food recommendation. In *The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18*, pages 146–162. Springer.

Ruibo Liu, Guoqing Zheng, Shashank Gupta, Radhika Gaonkar, Chongyang Gao, Soroush Vosoughi, Milad Shokouhi, and Ahmed Hassan Awadallah. 2022. Knowledge infused decoding. *arXiv preprint arXiv:2204.03084*.

George A. Miller. 1994. WordNet: A lexical database for English. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. Skill: Structured knowledge infusion for large language models. *arXiv preprint arXiv:2205.08184*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. Qagnn: Reasoning with language models and knowledge graphs for question answering. *arXiv preprint arXiv:2104.06378*.

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models for question answering. *arXiv preprint arXiv:2201.08860*.