Creating Synergy for Knowledge Infusion in Deep Learning for Question Answering: A Survey

Ankush Agarwal, Pushpak Bhattacharyya
IIT Bombay
{ankushagrawal, pb}@cse.iitb.ac.in

Abstract

LLMs often lack efficient domain-specific understanding, which is particularly crucial in specialized fields such as aviation. This necessitates the integration of knowledge graphs with deep learning which is thriving in improving the performance of various natural language processing (NLP) tasks. Knowledge graphs have emerged as a powerful framework for representing structured knowledge and capturing the relationships between entities in various domains. They provide a comprehensive and interconnected view of information, enabling efficient reasoning, search, and data integration. Language models are trained on a lot of data and perform well on tasks from the general domain. This survey paper presents an overview of the knowledge graph, deep learning, and the synergy of both knowledge graph and language models for question answering.

1 Introduction

There is a vast amount of online text available, presenting opportunities for productive utilization. In recent years, researchers have introduced two innovative methods to leverage this text effectively: 1) The concept of a Knowledge Graph emerged, aiming to represent text in a structured format. However, it became apparent that structuring all available free-form data posed significant challenges. 2) This limitation in Knowledge Graphs led to the emergence and widespread adoption of Deep Learning, a prominent technique in Machine Learning and Natural Language Processing. Deep Learning models are trained on textual data and applied across various applications. In our work, we leverage both Knowledge Graphs and Deep Learning for Information Extraction and Question Answering (QA) tasks.

Large language models (LLMs) have demonstrated remarkable performance in a wide range of natural language tasks. However, as these models continue to grow in size, they face significant challenges in terms of computational costs. Additionally, LLMs often lack efficient domain-specific understanding, which is particularly crucial in specialized fields such as aviation and healthcare. In this research, we emphasize the significance of incorporating knowledge bases into language models and propose a novel framework for integrating structured knowledge into smaller models. This approach addresses the challenges of computational efficiency while enabling effective domain-specific understanding.

2 Motivation

Building question-answering (QA) systems that can reason over knowledge graphs like AviationKG is challenging due to the scarcity of domain-specific datasets. While Large Language Models (LLMs) have shown impressive performance in various NLP tasks, their computational requirements and reliance on generic datasets limit their suitability for domain-specific tasks. Knowledge graphs (KGs) are valuable for representing and reasoning over knowledge but often suffer from incompleteness and noise, making it difficult to answer complex questions that require combining multiple pieces of information. To address these limitations, knowledge infusion is a technique that enhances the performance of deep learning models on KG-based QA tasks. It involves integrating knowledge from KGs into DL models, enabling them to learn improved representations and reason more effectively over the knowledge.

3 Knowledge Graph

Information can be effectively represented in the form of a graph consisting of nodes and edges. In this context, nodes correspond to the entities extracted from a given corpus, while edges denote the relationships between these entities within the graph structure. There are two main structures for constructing a knowledge graph:

1. Extracting triplets comprising a subject, predicate, and object from raw data. These triplets represent meaningful relationships between entities. Once the triplets are extracted, they are inserted into a graph database, which serves as the foundation for the knowledge graph.

2. Building the ontology, which defines the structure, hierarchy, and types of entities within the knowledge graph. Next, linking individuals within the knowledge graph using appropriate relations.

3.1 Ontology

Ontology has become popular in the Knowledge community. (Noy et al., 2001) states that ontology is a description of concepts in the discourse domain (called classes or concepts). The features of Ontology are:

- Each class has different features and certain restrictions.
- Ontology is dynamic and domain-centric.
- Ontology is formed with the help of taxonomy.

3.2 Taxonomy

Taxonomy is the process of categorizing objects into various classes and subcategories. It enables us to visually represent the hierarchical structure of classes, objects, and their properties. Unlike ontology, taxonomy is considered static as it does not possess the dynamic nature associated with ontology.

3.3 Entities and Relations

- Entity is a node in a Knowledge graph or an instance of a class in ontology. An entity can be a person, organization, etc., *i.e.*, an individual of a class.
- Relation is an edge in a Knowledge graph connecting two nodes or entities.

For example, "Raj *lives in* India". **lives in** is a relation between a person and a place.

3.4 Object Property and Data Property

In a knowledge base, object and data properties are two types of properties used to describe relationships between entities or assign attributes to entities.

- Object Property: An object property represents a relationship between two entities or individuals in a knowledge base. It denotes how entities are connected or related to each other.
  Ex: Milk *isA* Dairy Product. **isA** relation between Milk and Dairy Product.
  Object properties typically link two individuals together, indicating a connection or association between them.
- Data Property: A data property is used to assign attribute values or characteristics to entities in a knowledge base. It describes the properties or attributes of an individual entity.
  Ex: Milk *hasFat* 10grams. Here, **hasFat** is a data property.
  Data properties are used to store and represent data values such as numbers, strings, dates, or booleans.

3.5 Graph Database

A graph database stores nodes and relationships for the visual representation in the form of a graph. In our project, we have used two graph databases: Protégé and Neo4j. A short explanation for both databases is provided below.

Protégé

- Protégé is an open source software used for the construction of ontology, developed by Stanford Community.
- Protégé is supported by academic, government, and corporate users.
- Protégé works as an interface for the development of ontology containing multiple plugins such as Cellfie used for inserting data into a knowledge graph, SPARQL plugin, for retrieving results from an ontology using SPARQL queries.

Neo4j

- Neo4j is an open-source, NoSQL, and native graph database.
- Neo4j is used today by thousands of startups, educational institutions, and large enterprises in all sectors including financial services, government, energy, technology, retail, and manufacturing.

- Neo4j uses Cypher language, a declarative query language similar to SQL, but optimized for graphs.

We utilized Protégé to develop a Knowledge Graph (KG) specifically focused on aircraft accident reports sourced from the National Transportation Safety Board (NTSB). However, during our usage of SPARQL queries, we encountered limitations in retrieving information from the KG. In order to address these challenges and enhance our KG construction process, we compared various state-of-the-art (SOTA) graph databases and determined that Neo4j offers the most favorable features.

3.6 Different Types of Knowledge Graphs

A knowledge graph is a powerful tool used in various domains to represent and organize structured knowledge about the world. While the concept of a knowledge graph remains the same across different implementations, there can be variations in terms of their construction, scope, and focus. Here are some notable types of knowledge graphs.

1. Freebase: Freebase (Bollacker et al., 2008), developed by Metaweb and later acquired by Google, was one of the earliest and most well-known knowledge graphs. It aimed to create a comprehensive repository of structured information about entities, their attributes, and their relationships.

2. WikiData: WikiData (Vrandečić and Krötzsch, 2014) is a collaborative, multilingual knowledge graph project that serves as a central hub for structured data on a wide range of topics. It was initiated by the Wikimedia Foundation, the same organization behind Wikipedia, with the goal of providing a common data repository that can be used to enhance the information available in Wikipedia articles and other Wikimedia projects.

   WikiData follows a collaborative approach, inviting contributions from a global community of editors who add, edit, and curate the data. The project is multilingual, meaning it supports data in multiple languages, enabling the representation of knowledge from diverse cultures and regions.

   The structure of WikiData revolves around the concept of "items" and their associated "statements." An item in WikiData represents a specific concept or entity, such as a person or a place. Each item is assigned a unique identifier, known as a QID, to ensure consistency and enable cross-referencing with other items. Statements, on the other hand, capture specific properties or attributes of an item, along with their corresponding values. For example, an item about a person might have statements about their date of birth, occupation, nationality, and so on.

3. ConceptNet: ConceptNet (Liu and Singh, 2004) is a freely available knowledge graph that emphasizes the representation of general knowledge and common-sense reasoning. It contains a vast collection of assertions in the form of "concept A is related to concept B" and covers various aspects of human understanding.

4. AviationKG: An Aviation Knowledge Graph (Agarwal et al., 2022b) is created using NTSB reports and ADREP taxonomy in the Protégé tool. The construction process involves several steps, including pre-processing the NTSB reports, creating an ontology, building the Knowledge Graph, evaluating the KG, and addressing challenges encountered while creating the Aviation KG. Figure 1 illustrates the pipeline for constructing the KG from NTSB reports. A total of 4000 NTSB reports spanning from 1962 to 2015, with an average length of 3000 words per report, are utilized in this construction. These reports contain unstructured paragraphs and structured tables providing information about aircraft accidents.

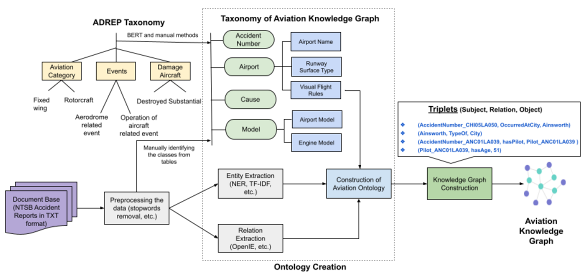

ADREP Taxonomy
BERT and manual methods
Aviation Category
Events
Damage Aircraft
Fixed wing
Rotorcraft
Aerodrome related event
Operation of aircraft related event
Destroyed
Substantial
Taxonomy of Aviation Knowledge Graph
Accident Number
Airport
Cause
Model
Airport Name
Runway Surface Type
Visual Flight Rules
Airport Model
Engine Model
Manually identifying the classes from tables
Document Base (NTSB Accident Reports in TXT format)
Preprocessing the data (stopwords removal, etc.)
Entity Extraction (NER, TF-IDF, etc.)
Relation Extraction (OpenIE, etc.)
Construction of Aviation Ontology
Ontology Creation
Triplets (Subject, Relation, Object)
(AccidentNumber_CHI05LA050, OccurredAtCity, Ainsworth)
(Ainsworth, TypeOf, City)
(AccidentNumber_ANC01LA039, hasPilot, Pilot_ANC01LA039)
(Pilot_ANC01LA039, hasAge, 51)
Knowledge Graph Construction
Aviation Knowledge Graph

Figure 1: Aviation Knowledge Graph Construction Process from NTSB reports

4 Deep Learning

Deep Learning is a specialized branch of machine learning that focuses on algorithms inspired by the architecture and functioning of the brain, specifically artificial neural networks.
In this section, we will look into the DL models, their training & fine-tuning process, and their applications which are useful in today's world.

4.1 Deep Learning Models

Deep learning models have gained extensive popularity due to their ability to extract high-level abstract features, deliver enhanced performance compared to traditional models, improve interpretability, and effectively process biological data. In the following section, we will examine three significant deep learning models utilized in our project: BERT, T5, and GPT-3.

BERT
BERT (Bidirectional Encoder Representations from Transformers), as described in the source (Devlin et al., 2019), is essentially a trained stack of Transformer Encoders. The primary technical advancement of BERT lies in its utilization of bidirectional training from the Transformer model, which is a widely-used attention-based model, for language modeling purposes.

How BERT Works ?
The BERT architecture is constructed based on the Transformer model. Presently, there exist two variants of BERT:

- BERT Base: 12 layers (transformer blocks), 12 attention heads, and 110 million parameters.

- BERT Large: 24 layers (transformer blocks), 16 attention heads, and 340 million parameters.

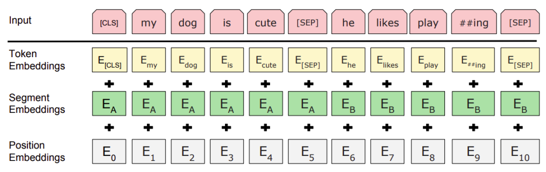

Input
[CLS] my dog is cute [SEP] he likes play ##ing [SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 2: BERT input representation. The input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings.

The developers behind BERT have added a specific set of rules to represent the input text for the model. Many of these are creative design choices that make the model even better.
The input embedding is a combination of 3 embeddings (shown in figure 2):

- Position Embeddings:To overcome the limitation of the Transformer model in capturing sequence and order information, BERT employs positional embeddings. These embeddings are learned and utilized by BERT to represent the position of words within a sentence. By incorporating positional embeddings, BERT effectively addresses the Transformer's inability to inherently capture sequential relationships, which is a strength of recurrent neural networks (RNNs).

- Segment Embeddings: BERT has the capability to process sentence pairs as inputs, which is particularly useful for tasks like question-answering. Segment embeddings play a crucial role in enabling BERT to learn distinct embeddings for each sentence in the pair. This allows the model to differentiate between the embeddings of different sentences.

  In Figure 2, the tokens labeled as EA correspond to the tokens belonging to sentence A, while the tokens labeled as EB correspond to those from sentence B. By incorporating segment embeddings, BERT can effectively capture the contextual information and relationships between the sentences in a pair, enhancing its ability to perform tasks that involve multiple sentences.

- Token Embeddings: BERT refers to the learned representations of individual tokens within a sentence. These embeddings capture the semantic and contextual information of each token in the input text. In BERT, token embeddings are generated by encoding each token in the input sequence. These embeddings capture the meaning and characteristics of the individual tokens, taking into account the surrounding context. By leveraging the large-scale pre-training process, BERT learns rich and contextualized representations for each token.

We will study the BERT's training and fine-tuning in the next section.

T5

T5 (Text-to-Text Transfer Transformer) is a model that aims to unify various downstream natural language processing (NLP) tasks into a consistent text-to-text format. Unlike BERT, T5 employs a single model architecture, loss function, and set of hyperparameters across all NLP tasks.

In T5, the inputs are structured in a manner that allows the model to recognize the specific task, while the output is generated as the textual representation of the desired outcome. This approach enables T5 to handle a wide range of NLP tasks by framing them as text generation problems.

The architecture of T5 is based on an Encoder-Decoder Transformer model. The encoder configuration in T5 shares similarities with BERT, but with modifications to support the text-to-text format.

By employing a consistent framework across tasks, T5 simplifies the training and deployment process for various NLP applications, offering a versatile and efficient approach to handling diverse NLP tasks.

GPT-3

GPT-3 (Generative Pre-trained Transformer 3) (Brown et al., 2020), is a deep learning-based language model that is capable of generating human-like text. It has the ability to generate various types of content such as code, stories, poems, and more. GPT-3 was introduced by OpenAI in May 2020 as an evolution of their previous language model, GPT-2.

GPT-3 is a highly advanced and large-scale language model. Given an input text, it utilizes deep learning techniques to probabilistically predict the tokens from a predefined vocabulary that are likely to follow. Language models like GPT-3 are statistical tools that provide predictions for the next word(s) in a given sequence. They essentially represent probability distributions over sequences of words.

With its sophisticated architecture and extensive pre-training, GPT-3 has demonstrated remarkable capabilities in generating coherent and contextually relevant text across various domains, making it a powerful tool for natural language generation tasks.

4.2 Training and Fine-tuning

Training is the initial process in which a model or architecture learns from labeled data to acquire knowledge and improve its performance. During training, the model undergoes optimization through iterations, adjusting its parameters to minimize the loss function and improve its ability to make accurate predictions.

Fine-tuning, on the other hand, is a subsequent process that is performed on top of a pre-trained model. Fine-tuning involves adapting the already trained model to a specific task or domain by further training it on task-specific or domain-specific labeled data. This process helps the model to specialize and adapt its learned representations to better suit the specific requirements of the target task or domain.

Now, let's explore how training and fine-tuning are applied to BERT, T5, and GPT-3:

BERT

BERT is pre-trained on two NLP tasks (refer Figure 3):

- Masked Language Modeling (MLM) - Let's say we have a sentence – "I love to read books on Kindle". We want to train a bi-directional language model. Instead of trying to predict the next word in the sequence, we can build a model to predict a missing word from within the sequence itself.
  Let's replace "Kindle" with "[MASK]". [MASK] is a token to denote that a word is missing. We'll then train the model in such a way that it should be able to predict "Kindle" as the missing token: "I love to read books on [MASK]."

- Next Sentence Prediction (NSP) - BERT is trained on the task of Next Sentence Prediction for tasks that require an understanding of the relationship between sentences. QA task is one such example. The task with NSP is: Given two sentences – A and B, is B the actual next sentence that comes after A in the corpus, or just a random sentence? Since it is a binary classification task, the data can be easily generated from any corpus by splitting it into sentence pairs.

This is how BERT is able to become a true task-agnostic model. It combines both the Masked Language Model (MLM) and the Next Sentence Prediction (NSP) pre-training tasks.
BERT can be fine-tuned on a variety of language

tasks such as Classification, QA, and Named Entity Recognition, by applying MLM and NSP.

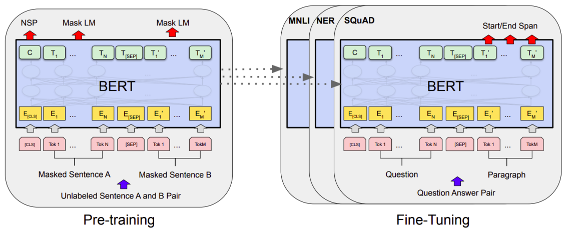


Figure 3: Overall pre-training and fine-tuning procedures for BERT.

T5

Training T5 involves training the model on a diverse set of tasks using a text-to-text format. The model is trained with a shared encoder-decoder architecture and a common loss function across different NLP tasks. The T5 model is trained on C4 - Colossal Clean Crawled Corpus. C4 is obtained by scraping web pages and ignoring the markup from the HTML. Fine-tuning T5 is done by further training the pre-trained model on task-specific labeled data, allowing it to specialize for specific tasks within the text-to-text framework.

GPT-3

The first thing that GPT-3 overwhelms with is its sheer size of trainable parameters which is 10x more than any previous model out there. In general, the more parameters a model has, the more data is required to train the model. As per the creators, the OpenAI GPT-3 model has been trained with about 45 TB of text data.
OpenAI, by default, gives us a few AI models or engine APIs, that are suited for different tasks such as QA, sentiment analysis, etc.

Both training and fine-tuning are essential processes in developing powerful language models like BERT, T5, and GPT-3, enabling them to learn from data and adapt their representations to various tasks or domains.

5 Question Answering System

Question answering (QA) is indeed a crucial problem in the field of natural language processing (NLP) and has been a longstanding milestone in artificial intelligence. QA systems enable users to pose questions in natural language and receive concise and immediate responses, typically encountered in search engines or information retrieval systems.

The goal of QA systems is to understand the user's query, process it, and provide relevant and accurate answers from a given knowledge base or corpus. These systems employ various techniques, including information retrieval, natural language understanding, and machine learning, to extract the most relevant information and generate appropriate responses to user queries.

QA systems have a wide range of applications, from general-purpose search engines to domain-specific question-answering in areas such as medicine, customer support, and education. They aim to bridge the gap between human language understanding and computational systems, enabling users to obtain specific and concise answers to their questions in a human-like manner.

5.1 QA Categorization

Figure 4 presents a categorization of QA systems into different types. The classification is based on four main aspects: Type of Question, Answer Type, Answer Source, and Modeling Approach.

Type of Question: Questions can take various forms, such as multiple-choice questions, conversational dialogues, visual format questions, or general inquiries. In our project, the focus is primarily on conversational questions, and we are developing a system specifically tailored to handle this type of question.

Answer Type: Answers in QA systems can be in the form of concise facts, relevant paragraphs, or a combination of both. In our project, we are using a hybrid approach where the answer type can vary based on the specific question and context.

Answer Source: QA systems can retrieve answers or relevant paragraphs from different sources, including a constructed Knowledge Graph (KG) or free-form text data. In our project, we utilize both the information stored in the Knowledge Graph and raw text data for question-answering purposes.

Modeling Approach: QA systems can be built using various approaches such as machine learning (ML), deep learning (DL), rule-based methods, or a combination of these techniques. In our project, we employ a rule-based approach and leverage natural language processing (NLP) techniques for constructing the Knowledge Graph and KGQA (Knowledge Graph Question Answering) system. Additionally, DL models are utilized to retrieve answers and relevant paragraphs from raw text data.

By considering these different aspects, our project aims to develop a comprehensive QA system that can handle conversational questions, provide hybrid answers, retrieve information from both the Knowledge Graph and text sources, and utilize a combination of rule-based and DL approaches to enhance the accuracy and effectiveness of the system.

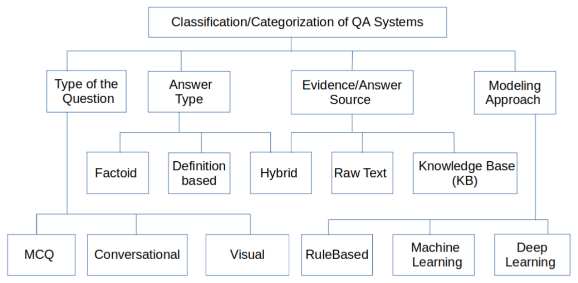


Figure 4: Question-Answering: Categorization

5.2 Popular Datasets

There are largely available QA datasets, generated either by crowd-sourcing or by manual annotation. Some of the popular datasets are listed below:

- The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), is a widely used reading comprehension dataset. It consists of question-answer pairs that are created through crowd-sourcing, utilizing Wikipedia as the primary source of information. SQuAD is available in two versions: SQuAD1.1 and SQuAD2.0. SQuAD1.1 contains a total of 107,785 question-answer pairs, while SQuAD2.0 is larger, containing 161,560 question-answer pairs.

  The dataset is designed to evaluate and train models for reading comprehension and question answering tasks. It provides a diverse range of questions covering various topics, allowing researchers and developers to assess the performance of their question-answering systems on real-world data. SQuAD has played a significant role in advancing the field of natural language processing and has been instrumental in the development and evaluation of numerous question answering models and techniques. Its availability and large-scale nature make it a valuable resource for training and evaluating QA systems.

- The WikiQA dataset (Yang et al., 2015), is a collection of 3,047 questions. These questions were extracted from Bing query logs using simple heuristics that involved identifying queries beginning with WH-questions (such as "what," "who," "where," etc.) and ending with a question mark ('?'). The purpose of the WikiQA dataset is to provide a resource for training and evaluating question-answering systems. The dataset contains a diverse set of questions that users typically ask in search engines, making it relevant for developing models that can understand and respond to real-world user queries.

- The NewsQA dataset (Trischler et al., 2016), is a collection of 119,633 natural language questions. These questions were generated by crowd-workers and are based on 12,744 news articles sourced from CNN. In the NewsQA dataset, the crowd-workers also highlighted the specific text spans within the news articles that contain the answers to the corresponding questions. This allows for the development and evaluation of question-answering systems that can accurately locate and extract relevant information from news articles.

There are many more publicly available QA datasets.

5.3 Evaluation Metrics

There are many evaluation methods available in the field of QA.

- Precision, Recall, F1 Score: For factoid-based question-answering, the standard way to measure performance is to calculate precision, recall, or F1 score.
  Precision measures the proportion of correctly predicted positive instances, *i.e.*, correct answers out of all instances predicted as positive. It quantifies the system's ability to provide accurate and relevant answers. Precision is calculated as the ratio of true positives (correctly predicted answers) to the sum of true

positives and false positives (incorrectly predicted answers).
Recall, on the other hand, measures the proportion of correctly predicted positive instances out of all actual positive instances, *i.e.*, all correct answers. It captures the system's ability to retrieve all the relevant answers. The recall is calculated as the ratio of true positives to the sum of true positives and false negatives (missed answers).
The F1 score combines both precision and recall into a single metric, providing a balanced evaluation of the system's performance. It is the harmonic mean of precision and recall and takes into account both the precision and recall values. The F1 score is calculated as 2 times the product of precision and recall, divided by the sum of precision and recall.

These metrics, precision, recall, and F1 score, help in assessing the accuracy, completeness, and overall performance of factoid-based question answering systems, providing insights into how well they are able to provide correct and relevant answers to the given questions.

- BLEU: Bilingual Evaluation Understudy (BLEU) is an evaluation metric commonly used in machine translation tasks, but it can also be applied to other natural language processing tasks, including question answering. BLEU measures the similarity between a candidate response (generated by a system) and one or more reference responses (provided by human evaluators). It calculates a modified n-gram precision by comparing the n-grams (contiguous sequences of words) in the candidate response with those in the reference responses.

- ROUGE: It is a set of evaluation metrics commonly used for assessing the quality of automatic summarization, machine translation, and question-answering systems. ROUGE-1, ROUGE-2, and ROUGE-N are the most widely used variants of the ROUGE metrics. ROUGE-1 measures the overlap of unigrams (individual words), ROUGE-2 measures the overlap of bigrams (pairs of consecutive words), and ROUGE-N measures the overlap of n-grams (contiguous sequences of n words).

6 Dataset

This section provides a comprehensive explanation of the dataset creation process to facilitate research on knowledge infusion into deep learning models for question-answering. The created dataset contains AviationKG, and question-answering datasets, *i.e.*, AviationQA, and AeroQA.

6.1 AviationKG

We create an Aviation Knowledge Graph using NTSB reports and ADREP taxonomy in the Protégé tool. The construction process involves several steps, including pre-processing the NTSB reports, creating an ontology, building the Knowledge Graph, evaluating the KG, and addressing challenges encountered during the creation of the Aviation KG. Figure 1 illustrates the pipeline for constructing the KG from NTSB reports. A total of 4000 NTSB reports spanning from 1962 to 2015, with an average length of 3000 words per report, are utilized in this construction. These reports contain both unstructured paragraphs and structured tables that provide information about aircraft accidents.

Pre-processing
We preprocess the NTSB reports before proceeding with entity and relation extraction. First, the NTSB reports are converted from PDF to TXT files, followed by the application of techniques like stopword-removal, PoS tagging, and lemmatization.

Ontology Creation
With the help of domain experts and ADREP taxonomy, we construct an ontology from the accident reports. For example, domain knowledge from ADREP 'Events' taxonomy is used for creating the *Event* class. 'Events' taxonomy is relevant because every accident report has an event sequence that gives a gist of the cause. For ontology creation, we extract the ADREP taxonomy in a tree-like data structure such that a unique path exists from the root to each leaf node. Subsequently, we obtain classes by mapping NTSB occurrences to the adequate root-to-leaf paths.
Following is an example of an ADREP event mapped to a NTSB occurrence.

Sample root-to-leaf path in ADREP Taxonomy
Aircraft Events
Operation of the aircraft related event
Aircraft handling related event
Dragged wing/rotor/pod/float

ADREP 'Events' taxonomy contains a number of events and its root node is 'Aircraft Events'. An event involving a 'dragging of a wing' is under the category of 'Aircraft handling'. This root-to-leaf path of 'Dragged wing/rotor/pod/float', ADREP event is mapped to 'DRAGGED WING, ROTOR, POD, FLOAT OR TAIL/SKID', which occurs as an event in a NTSB report.

However, ADREP taxonomy is not always compatible with the NTSB documents for creating classes. Hence, we apply the following extraction techniques to the NTSB text for finding entity classes and their instances:

- **Named Entity Recognition (NER)** is used to identify names, organizations, etc., from the NTSB text. These named entities are inserted as entity classes in our ontology. For example, Long Beach, Las Vegas, and Chicago are the instances of *Location* class present in NTSB reports identified by the NER method.

- **Term Frequency – Inverse Document Frequency (TF-IDF)** technique is used to determine the important terms across NTSB documents, which are later organized as entity classes in ontology. For example, Aircraft ID, Aircraft Damage, and Pilot Certificate are some of the important terms identified by the TF-IDF technique.

- **C-value** overcomes the drawback of TF-IDF in obtaining multi-word terms. Multi-words are necessary to identify the entities such as Landing Gear, Vertical Stabilizers, etc., in aircraft reports.

- **Sentence Clustering** is used for grouping similar sentences and discovering the entities among them.

- **Syntactic Analysis** technique is used where the corpus is annotated with part-of-speech tags to find hypernyms, hyponyms, and meronyms. We extract all the sentences containing three or more nouns in the analysis. Then, we manually try to identify lexically related entities. The intuition behind selecting nouns is that most of the time, the entities (classes and instances) are tagged as nouns in the ontology. Examples of such findings are – (a) 'Scratches are found on engine's nacelle.' From this, we can observe that *nacelle* is $\underline{\text{part of}}$ an *engine*. (b) 'During the engine inspection, it was observed that the wing mounted engine was working, but the fin mounted engine was not working.' Here, we observe that *wing mounted engine* and *fin mounted engine* are $\underline{\text{types of}}$ *engine*.

- **DL based techniques** are also used for entity extraction such as NER_DL, Onto_100 and NER_DL_BERT (trained on CoNLL dataset and uses BERT embeddings).

Dependency Analysis and **Open-IE** techniques are used for relation extraction. The extracted relations are observed and inserted as properties in our Aviation Ontology. Furthermore, we manually add some properties by observing entity classes in ontology from the NTSB reports.

Knowledge Graph Construction

We have an Aviation Ontology with entity classes, instances, object properties, and data properties. The entities and relations must be linked in the form of triples. We look at entities and relations in the constructed ontology to extract triples from the NTSB reports using regular expressions. These extracted triples are inserted into an ontology to form a Knowledge Graph.

Metrics	**Count(#)**
Entity Class	239
Individual	8894
Object Property	300
Data Property	71
Axioms	97879
Part of Relation among Classes	494
Property of Relation among Classes	353

Table 1: Properties of Aviation Knowledge Graph: Aviation KG is constructed in Protégé where the class count and instances are displayed.

Table 1 describes the properties for Aviation Knowledge Graph constructed from the NTSB reports in Protégé.

Knowledge Graph Evaluation

We manually evaluate the terms obtained through entity and relation extraction techniques. A domain expert in Honeywell Corporation provided a small set of cases to validate the reach of the constructed Knowledge Graph. A total of 120 SPARQL queries with gold answers of different categories were tested, where our KG answered 83 questions, thereby achieving an accuracy of 69.1%.

6.2 Aviation Corpus: A dataset consisting of Aviation text

The MetaQA dataset requires a C4 corpus for the MLM training with the SKILL approach. To conduct experiments using our AeroQA dataset, we compiled the Aviation corpus, comprising 665k lines of English text related to the aviation domain. This corpus was obtained by scraping 4,000 National Transportation Safety Board reports from the NTSB website, covering the period between 1981 and 2018. The reports, initially in PDF format, were converted to JSON format for easier processing. The paragraphs that contain clean text were extracted from selected sections of the reports which are Analysis, Probable Cause and Findings, and Factual Information. The selected paragraphs were then curated and included in the Aviation corpus, which served as a valuable resource for our research and experimentation.

6.3 Created QA dataset for knowledge infusion: AviationQA and AeroQA

In the context of knowledge infusion into language models for the aviation domain, we will discuss two carefully curated QA datasets, AviationQA and AeroQA. These datasets have been specifically designed to enhance the performance of language models in answering aviation-related questions.

AviationQA
A synthetic dataset, AviationQA (Agarwal et al., 2022a) is created for question-answering in the aviation domain. We web scrape the National Transportation Safety Board (NTSB) website and download 12k reports from 2009-2022. A set of 90 question templates is prepared using the common structure of documents in the format:

- Where did the accident [ ] take place?

- What is the model/series of the aircraft bearing accident number [ ]?

- Was there fire on the aircraft of the accident number [ ]?

The template of questions is created, and answers to those questions are extracted from every NTSB report. Because every report is associated with an accident number, we place [ ] in the template to indicate which report the question pertains to, e.g., CHI07LA273, LAX07LA148. NTSB reports are semi-structured, containing unstructured data in paragraphs and structured data in tabular format. We extract answers from each report w.r.t the template using the regular expression method. Later, QA pairs are scrutinized. As some reports' structure varies, different scripts are written to fetch answers for those reports.

We successfully created 1 million factoid QA pairs in the aviation domain using the template-based method. The dataset will contribute to research and development in the aviation industry.

AeroQA
To address the limitations of the AviationQA (Agarwal et al., 2022a) dataset and evaluate the reasoning ability over the AviationKG knowledge graph, we have created AeroQA, a multi-hop question-answering dataset. While AviationQA is a large dataset in the aviation domain, it is limited in two key aspects. Firstly, all the questions in AviationQA are single-hop, which does not allow for evaluating the model's ability to reason over knowledge graphs like AviationKG. Secondly, only a fraction of AviationQA pairs contain questions that can be answered using the triples from AviationKG, limiting the exploitation of the full reasoning potential of the QA pairs. AeroQA is specifically curated to overcome these limitations and provide a dataset that facilitates reasoning over KGs in the aviation domain.

AeroQA, a comprehensive dataset for the aeronautics domain, was constructed by scraping NTSB reports from the official NTSB website. This dataset complements the pre-processed AviationKG knowledge graph and enables reasoning tasks. AeroQA encompasses over 31k questions designed for both single-hop and multi-hop reasoning. The dataset is divided into three subsets: training, validation, and testing, with an 80:10:10 split ratio. Below, we present a selection of examples from the AeroQA dataset to provide a glimpse into its content.

Examples of Single-hop Questions in AeroQA:

- Q: What certificate does [Pilot_ATL03LA101] have?
  A: Private

- Q: What is the engine manufacturer associated with [Registration_N127RB]?
  A: Lycoming

- Q: What caused [AccidentNumber_FTW93LA202]?
  A: Pre-Flight Planning | Fluid Fuel | Terrain Condition

Examples of Two-hop Questions in AeroQA:

- Q: What is the aircraft category of the registered aircraft involved in [AccidentNumber_MIA05LA036]?
  A: Airplane

- What could have contributed to the cause of the accident [AccidentNumber_SEA96TA046]?
  A: Pilot in Command | Pilot of other Aircraft | Check Pilot

The AeroQA dataset includes multiple answers for each question, separated by the '|' symbol. There are 83 relations for the 1-hop QA pairs and 26 relations for the 2-hop QA pairs in the dataset. These relations serve as templates for constructing the QA pairs and were derived using the prompt-based approach with ChatGPT. The template structure examples of the AeroQA dataset are presented in Table 2 for 1-hop questions and Table 3 for 2-hop questions.

7 Knowledge Graph and Deep Learning based Question Answering Systems

In this section, our focus will be on exploring different hybrid Question Answering (QA) systems that integrate both Knowledge Graph (KG) and Deep Learning (DL) approaches. The construction of a hybrid system is a central concept driving our project. We will delve into the methodologies and techniques employed by these hybrid QA systems to leverage the strengths of both KG and DL. By combining the structured knowledge representation of a KG with the learning capabilities of DL models, these hybrid systems aim to enhance the accuracy, comprehensiveness, and efficiency of question-answering.

7.1 Hybrid Question Answering over Knowledge Base and Free Text

In paper (Xu et al., 2016), the authors introduce a hybrid question-answering (hybrid-QA) system that combines the use of a structured knowledge base (KB) and free text to answer questions. The motivation behind this approach is the inherent incompleteness of knowledge bases, which limits the scope of questions that can be answered solely based on structured data. To address this limitation, the authors propose an Integer Linear Program (ILP) model for hybrid-QA (Figure 5). This model leverages both the KB and free text to find the most relevant and accurate answer to a given question. The ILP model is designed to handle various types of questions and generate query templates that can be used to retrieve information from the KB and free text sources.

As an example, the paper presents the question "Who is the front man of the band that wrote Coffee TV?" This question serves as an illustration of how the ILP model can be applied to answer complex queries by combining information from the KB and free text sources. By studying the ILP model and its application in this hybrid-QA system, we can gain insights into the integration of structured and unstructured data sources for question answering.

The hybrid-QA system described in the paper follows a process that involves the decomposition of the question into triplets, the extraction of relations from both the knowledge base (KB) and the text, and the identification of entities from the KB and text based on the obtained relations. Here is a rephrased version of the steps:

1. The question is analyzed and decomposed into triplets consisting of subject (subj), relation (rel), and object (obj). This decomposition allows for entity linking, where possible, knowledge base entities mentioned in the question are identified.

2. Two types of relation extractors are used to predict both KB predicates and textual relations. These extractors aim to identify the relationships between entities mentioned in the question or between question words and entities.

3. The entities mentioned in the question are further identified by considering the predicates obtained from the KB. Similarly, the predicates obtained from the text are validated with

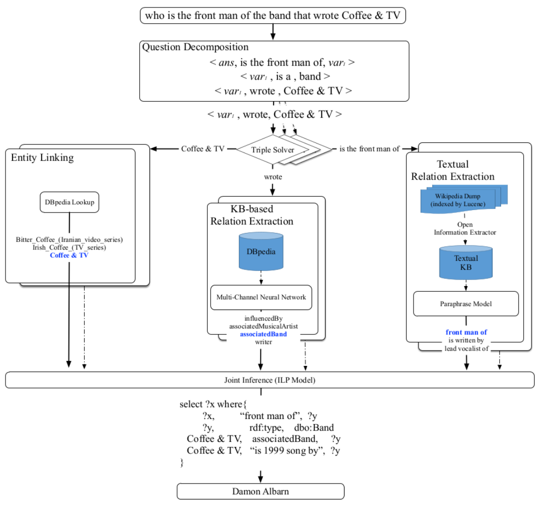
Relation	Template
hasAircraftManufacturer	What is the aircraft manufacturer associated with [HEAD]
hasFederalAviationRegulation	What is the Federal Aviation Regulation associated with [HEAD]
OccurredAtCountry	In which country did [HEAD] occur

Table 2: The table displays the templates employed in constructing the AeroQA 1-hop dataset. These templates utilize the placeholder [HEAD], which corresponds to the entity, *i.e.*, accident number, and registration number mentioned in the report.

Relation1	Relation2	Template
hasRegistrationNumber	hasAirworthinessCertificate	What is the airworthiness certificate of the registered aircraft involved in [HEAD]
IsCausedBy	IsCausedDueTo	What could have contributed to the cause of the accident [HEAD]
hasPilot	hasInstructorRating	What was the instructor rating of the pilot in the aircraft involved in [HEAD]

Table 3: The table showcases the templates used for constructing the AeroQA 2-hop dataset. In these templates, the placeholder [HEAD] represents the entity, *i.e.*, accident number, and registration number of the report, which is utilized to generate the 2-hop AeroQA pairs.

the entities mentioned in the question. This step helps in refining the understanding of the relationships between entities and the specific context of the question.


who is the front man of the band that wrote Coffee & TV
Question Decomposition
< *ans*, is the front man of, var_1 >
< var_1 , is a , band >
< var_1 , wrote , Coffee & TV >
< var_1 , wrote, Coffee & TV >
Coffee & TV
Triple Solver
is the front man of
wrote
Entity Linking
DBpedia Lookup
Bitter_Coffee_(Iranian_video_series)
Irish_Coffee_(TV_series)
Coffee & TV
KB-based Relation Extraction
DBpedia
Multi-Channel Neural Network
influencedBy
associatedMusicalArtist
associatedBand
writer
Textual Relation Extraction
Wikipedia Dump (indexed by Lucene)
Open Information Extractor
Textual KB
Paraphrase Model
front man of
is written by
lead vocalist of
Joint Inference (ILP Model)
Damon Albarn

```
select ?x where{
      ?x,         "front man of",  ?y
      ?y,            rdf:type,     dbo:Band
  Coffee & TV,   associatedBand,     ?y
  Coffee & TV,   "is 1999 song by",  ?y
}
```

Figure 5: A running example of the hybrid-QA system for the question *who is the front man of the band that wrote Coffee & TV*, where the blue annotations are correct.

By performing these steps, the hybrid-QA system combines the information from the knowledge base and the text to effectively answer questions that require both structured and unstructured data sources.

7.2 K-BERT: Enabling Language Representation with Knowledge Graph

K-BERT (Liu et al., 2020), as described in the paper, utilizes a technique where triples from a knowledge graph (KG) are integrated into sentences as domain knowledge. The architecture of K-BERT is depicted in Figure 6.

In K-BERT, a sentence tree is constructed by leveraging the KG and the input question. The sentence tree consists of triplets connected from the KG, with entities that are matched to the entities mentioned in the input question. This sentence tree is then passed through an embedding layer and a visible layer, as illustrated in Figure 7. The embedding layer converts the sentence tree into token-level representations, including token embeddings, position embeddings, and segment embeddings. These representations capture the contextual information and the relationships between the entities in the sentence tree. The visible layer, on the other hand, plays a crucial role in addressing the knowledge noise (KN) issue. By utilizing the visible matrix, the visible layer prevents semantic changes in the sentence representation. For example, in Figure 7, the relation between "China" and "Apple" is faded in the embedding representation since "China" is not directly related to "Apple". This helps mitigate the potential influence of excessive knowledge in sentence representation.

The knowledge-rich encoding obtained from K-BERT can be used for various purposes, such as classification and question-answering tasks, where incorporating domain knowledge from the KG is beneficial. By integrating triples from the KG into the sentence representation while managing knowledge noise, K-BERT enhances the model's understanding and performance on tasks that require both textual information and domain-specific knowledge.

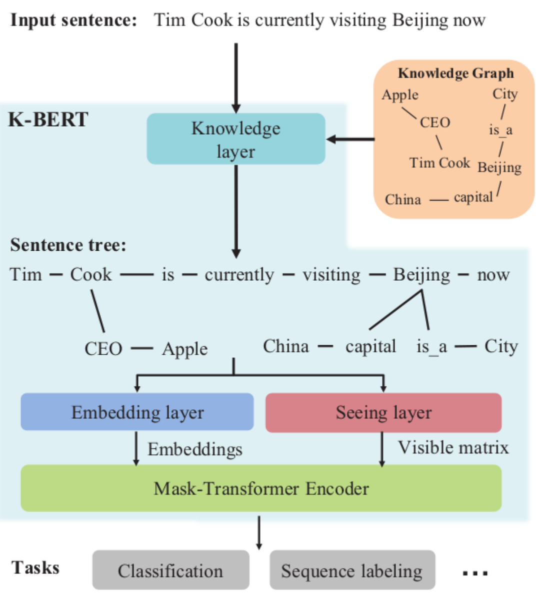


Figure 6: The model structure of K-BERT

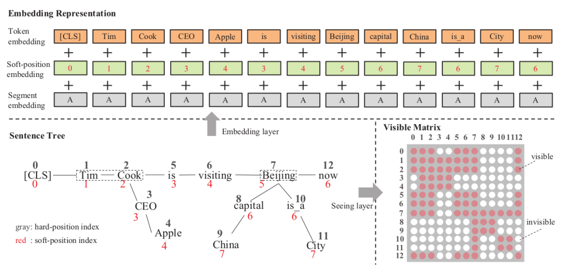


Figure 7: The process of converting a sentence tree into an embedding representation and a visible matrix.

7.3 SRLGRN: Semantic Role Labeling Graph Reasoning Network

SRLGRN (Zheng and Kordjamshidi, 2020), as presented in the paper, is a graph reasoning network that leverages the semantic structure of sentences to learn cross-paragraph reasoning paths and identify supporting facts and answers jointly. The architecture of SRLGRN, depicted in Figure 8, consists of several components: Paragraph Selection, Graph Construction, Graph Encoder, Supporting Fact Prediction, and Answer Span Prediction.

In the Paragraph Selection step, given a question as input, the BERT model is employed to select the most relevant two paragraphs from a set of n paragraphs. These selected paragraphs serve as the basis for subsequent processing. Next, the Graph Construction step involves creating a sub-graph using the question and the two selected paragraphs. The selected paragraphs are passed through an encoder to generate token and sentence embeddings. These embeddings, along with the question, contribute to constructing the sub-graph. The constructed sub-graph is then encoded using a Graph Convolutional Network (GCN) in the Graph Encoder step. The GCN incorporates information from the graph structure and captures the relationships between graph elements, facilitating reasoning over the sub-graph. For the task of Supporting Fact Prediction, the graph sentences and sentence embeddings are concatenated. This concatenated representation is used to predict the supporting facts, which are crucial pieces of information that provide evidence for the answer. Similarly, for Answer Span Prediction, the graph arguments and token embeddings are concatenated. This concatenated representation is utilized to predict the answer span, indicating the specific range of tokens within the context that contains the answer.

By employing graph reasoning and jointly considering supporting facts and answer span prediction, SRLGRN enhances the understanding and reasoning capabilities of the model, enabling it to effectively tackle complex question-answering tasks that require cross-paragraph comprehension and reasoning.

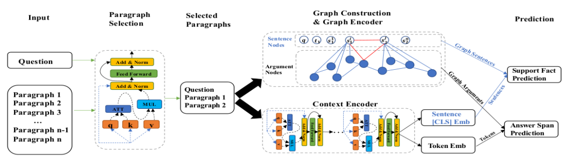


Figure 8: The Semantic Role Labeling Graph Reasoning Network (SRLGRN) model

7.4 KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation

Knowledge **E**mbedding and **P**re-trained **L**anguag**E** **R**epresentation (KEPLER) (Wang et al., 2021), as described in the paper, is a framework that aims to integrate factual knowledge into Pre-trained Language Models (PLMs) and generate effective text-enhanced knowledge embeddings. The architecture of KEPLER, illustrated in Figure 9, consists of several components: entity embedding, relation embedding, Masked Language Modeling (MLM) encoding, and training objectives.

In KEPLER, the entity descriptions in Knowledge Graph (KG) triplets are encoded to obtain entity embeddings. Similarly, relation embeddings are created to represent the relationships between entities in the KG. The MLM encoding is performed on the supporting text, leveraging the pre-trained PLM. This process involves masking certain tokens in the text and training the model to predict the masked tokens based on the surrounding context. By performing MLM encoding, KEPLER enhances the understanding and representation of the text.

Both the knowledge embedding and MLM objectives are trained on the same PLM, allowing the model to effectively incorporate factual knowledge and leverage the power of the PLM for language understanding. After the training process, the fine-tuned KEPLER model can be utilized for various applications, leveraging its integrated knowledge embeddings and enhanced language representation capabilities.

KEPLER offers a comprehensive framework that merges factual knowledge from Knowledge Graphs (KGs) with the powerful capabilities of Pre-trained Language Models (PLMs). This integration leads to enhanced knowledge integration and the generation of highly effective text-enhanced knowledge embeddings.

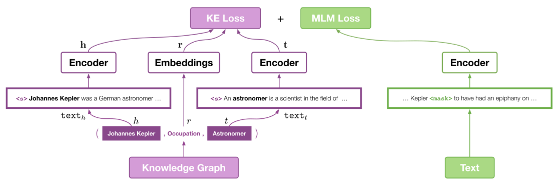

KE Loss
+
MLM Loss
h
r
t
Encoder
Embeddings
Encoder
Encoder
<s> Johannes Kepler was a German astronomer …
<s> An astronomer is a scientist in the field of …
… Kepler <mask> to have had an epiphany on …
text_h
h
r
t
text_t
(Johannes Kepler , Occupation , Astronomer)
Knowledge Graph
Text

Figure 9: The KEPLER framework

7.5 KGT5: Sequence-to-Sequence Knowledge Graph Completion and Question Answering

KGT5 (Saxena et al., 2022) presents a novel approach where both knowledge graph link prediction and question-answering tasks are framed as sequence-to-sequence (seq2seq) tasks. The framework of KGT5, depicted in Figure 10, involves training a T5 model from scratch using KG triplets and QA pairs. The T5 model is trained by extracting triplets (s, p, o) from the KG and converting them into verbalized forms such as (s, p, ?) and (?, p, o). This training process enables the T5 model to learn the patterns and relationships within the KG. Subsequently, the model is fine-tuned using QA pairs, resulting in the development of a question-answering system named KGT5.

Figure 11 illustrates a comparison between the link prediction approach used in conventional Knowledge Graph Embedding (KGE) and the approach employed in KGT5.

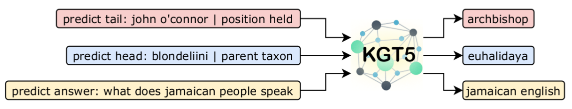

predict tail: john o'connor | position held
predict head: blondeliini | parent taxon
predict answer: what does jamaican people speak
KGT5
archbishop
euhalidaya
jamaican english

Figure 10: Overview of KGT5 method

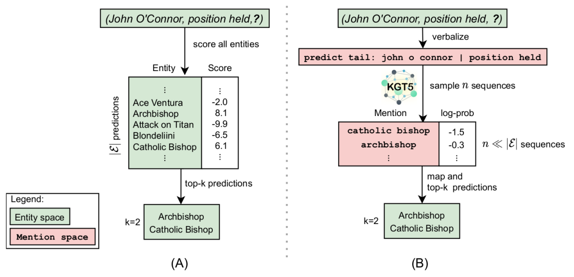


Figure 11: Inference pipeline of (A) conventional KGE models versus (B) KGT5 on the link prediction task.

7.6 SKILL: Structure Knowledge Infusion for Language Models

We investigate incorporating knowledge into language models (LMs) using knowledge triples and textual information. The flow diagram depicted in Figure 12 illustrates the adopted SKILL (Moiseev et al., 2022) approach. In this approach, triples are extracted from the knowledge graph (KG) and combined with the text to prevent any degradation in the model's performance on natural language understanding tasks. After combining the triples and text, the T5 model is continually pre-trained using a salient masked language modeling technique. This process results in a knowledge-infused T5 model. Subsequently, the trained model is fine-tuned for the specific task, which in this case is a question answering (QA), leading to the creation of the fine-tuned model.

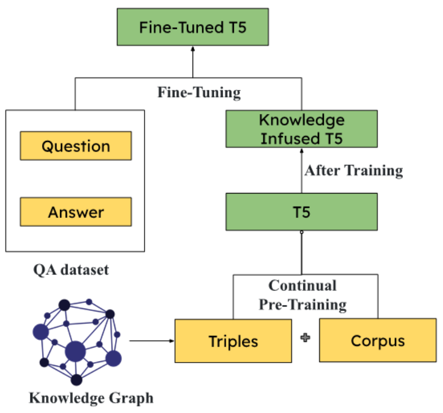


Figure 12: Illustrative figure represents the SKILL approach (Moiseev et al., 2022), which involves two steps. First, the model undergoes continual pre-training using triples+text. Subsequently, the model is fine-tuned for specific tasks, such as question answering (QA).

7.7 QA-GNN and GREASELM

In this section, we will explore two state-of-the-art architectures, namely QA Graph Neural Network (QA-GNN) (Yasunaga et al., 2021) and Graph Reasoning Enhanced Language Models (GREASELM) (Zhang et al., 2022), which leverage both Knowledge Graphs (KG) and Language Models (LM) for question answering and reasoning.

QA-GNN Model

- The QA-GNN (Yasunaga et al., 2021) approach combines a question and an answer to create a QA context node. A subgraph is extracted based on this QA context node, and a joint graph is formed by connecting the context node with entities from the QA context (see Figure 13).
- Language Models (LM) are utilized to obtain representations from the QA context, capturing the information in a comprehensive manner.
- LMs are employed to determine the importance of KG nodes in relation to the given QA context. Relevance scores are calculated using LM, facilitating the understanding of the relationship between the context node and other nodes in the graph.
- A Graph Neural Network (GNN) module is constructed to update node representations. It employs iterative message passing between neighboring nodes, allowing for the refinement and enhancement of representations (see Figure 14).

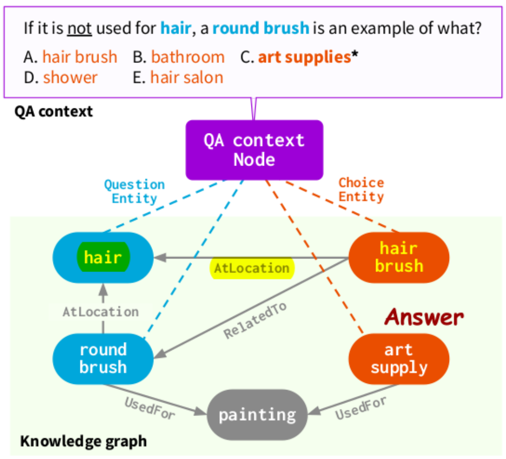


Figure 13: Combining QA context node to form the sub-graph.

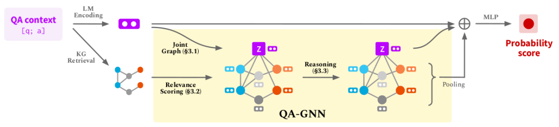


Figure 14: Overview of QA-GNN approach.

GREASELM

The GREASELM (Zhang et al., 2022) model, which is derived from QA-GNN, employs a comprehensive architecture for reasoning with KG and LM. The architecture, depicted in Figure 15, consists of the following key components and steps:

- Textual Context: The textual context, along with a special interaction token, is appended and passed through multiple layers of LM-based unimodal encoding. This encoding captures the textual information in a layered manner.
- Local KG Extraction: A relevant local KG is extracted, and an interaction node is connected to it. The KG entities interact indirectly with the tokens in the language context through the interaction node, enabling knowledge integration.
- GREASE LM Layers: In subsequent GREASE LM layers, both the language representation and the KG are updated. The language representation continues to be refined

through LM layers, while the KG undergoes processing using a Graph Neural Network (GNN), enabling reasoning over its knowledge.

- Modality Interaction (MInt): At each layer, the representations of the interaction token and node are pulled, concatenated, and passed through a Modality Interaction (MInt) unit. This unit mixes their representations, facilitating the fusion of knowledge from the KG and language context.

- Iterative Updates: In subsequent layers, the mixed information from the interaction elements combines with their respective modalities. This allows the knowledge from the KG to influence the representations of individual tokens, and the context from language to impact the fine-grained entity knowledge representations in the GNN.

Through this iterative process of updating representations and integrating knowledge from KG and language context, GREASELM enables effective reasoning for question answering tasks.

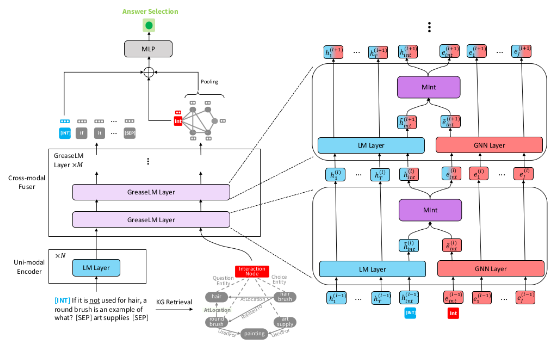


Figure 15: GREASELM Architecture

8 Summary

This survey paper offers a comprehensive overview of the use of knowledge graphs, deep learning, and knowledge infusion techniques in language models for question-answering tasks. It explores the fundamental aspects and characteristics of knowledge graphs, as well as the training and fine-tuning methodologies of deep learning models. The paper extensively examines question-answering, encompassing popular datasets and evaluation metrics employed in this domain. Furthermore, it conducts a thorough investigation of various knowledge infusion techniques, accompanied by the creation of knowledge graphs and question-answering datasets.

9 Conclusion and Future Research Directions

In conclusion, it is evident that the performance of a DLQA system is constrained without domain knowledge. However, incorporating domain-specific knowledge into the DL system yields significant improvements in the results. Integrating a pre-trained language model (LM) with domain-specific knowledge becomes crucial. This combination not only leverages the power of DL models but also harnesses the comprehensive understanding provided by the LM, resulting in a more robust and effective question-answering system. By bridging the gap between domain-specific knowledge and language understanding capabilities, this integrated approach holds great promise for advancing the field of question-answering and facilitating more accurate and insightful responses.

As a future research direction, it is crucial to acknowledge the limitations of current knowledge sources, including knowledge graphs, and the vast availability of language models. Therefore, our future efforts will focus on exploring techniques to directly integrate domain-specific knowledge into language models, potentially replacing conventional knowledge bases. This integration aims to optimize the utilization of knowledge in a more efficient and effective manner.

References

Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal*, 28(5):793–819.

Ankush Agarwal, Sakharam Gawade, Sachin Channabasavarajendra, and Pushpak Bhattacharya. 2022a. There is no big brother or small brother:knowledge infusion in language models for link prediction and question answering. In *Proceedings of the 19th International Conference on Natural Language Processing (ICON)*, pages 204–211, New Delhi, India. Association for Computational Linguistics.

Ankush Agarwal, Raj Gite, Shreya Laddha, Pushpak Bhattacharyya, Satyanarayan Kar, Asif Ekbal, Prabhjit Thind, Rajesh Zele, and Ravi Shankar. 2022b. Knowledge graph - deep learning: A case study in question answering in aviation safety domain. In *Proceedings of the Thirteenth Language Resources and*

Evaluation Conference, pages 6260–6270, Marseille, France. European Language Resources Association.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hugo Liu and Push Singh. 2004. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226.

Weijie Liu, Peng Zhou, Zhe Zhao, Zhiruo Wang, Qi Ju, Haotang Deng, and Ping Wang. 2020. K-bert: Enabling language representation with knowledge graph. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 2901–2908.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409, Austin, Texas. Association for Computational Linguistics.

Fedor Moiseev, Zhe Dong, Enrique Alfonseca, and Martin Jaggi. 2022. SKILL: Structured knowledge infusion for large language models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1581–1588, Seattle, United States. Association for Computational Linguistics.

Natalya F Noy, Deborah L McGuinness, et al. 2001. Ontology development 101: A guide to creating your first ontology.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, Dublin, Ireland. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2016. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*.

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

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. Kepler: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194.

Kun Xu, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Hybrid question answering over knowledge base and free text. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2397–2407.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 2013–2018.

Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. 2021. 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*, pages 535–546, Online. Association for Computational Linguistics.

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*.

Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J Smola, and Le Song. 2018. Variational reasoning for question answering with knowledge graph. In *Thirty-second AAAI conference on artificial intelligence*.

Chen Zheng and Parisa Kordjamshidi. 2020. Srlgrn: Semantic role labeling graph reasoning network. *arXiv preprint arXiv:2010.03604*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.