Knowledge Graph and Deep Learning Assisted Question Answering and Ontology Construction: A Survey

Sakharam Gawade, Pushpak Bhattacharyya
IIT Bombay
{sakharamg, pb}@cse.iitb.ac.in

Abstract

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. Knowledge graph infusion is a method to teach the domain to a language model and perform better on domain-specific tasks. While KGs from specialized domains are rare to obtain and difficult to create, many OpenIE techniques can accelerate the process. This survey paper presents an overview of knowledge graphs, focusing on their construction, representation, and utilization in QA for specialized domains.

1 Introduction

Knowledge graphs (KGs) are a powerful tool for representing and reasoning over knowledge. They can be used to answer questions, generate text, and make predictions. However, KGs are often incomplete and noisy. This can make it difficult to answer questions that require reasoning over multiple pieces of knowledge.

Deep learning (DL) has been shown to be effective for question answering (QA) tasks. However, DL models typically require large amounts of training data. This can be a challenge for KG-based QA tasks, as there is often limited labeled data available.

Knowledge infusion is a technique that can be used to improve the performance of DL models on KG-based QA tasks. Knowledge infusion involves injecting knowledge from a KG into a DL model. This can help the model to learn better representations of the knowledge and to reason more effectively over it.

There are a number of ways to infuse knowledge into a DL model. One common approach is to use a knowledge base embedding (KBE) technique. KBE techniques map entities and relations from a KG to a vector space. This allows the DL model to learn the meaning of the entities and relations in the KG.

Another approach to knowledge infusion is to use a knowledge guided attention mechanism. Knowledge guided attention mechanisms allow the DL model to focus its attention on the most relevant parts of the KG when answering a question.

This paper discusses the construction of KGs, KG embeddings, IR, and knowledge infusion into language models. It provides an overview of the state of the art in these areas and identifies some of the challenges that remain in domain-specific knowledge infusion.

2 Motivation

Domain specific datasets are scarce and highly sought after, posing challenges for building question-answering (QA) systems capable of reasoning over knowledge graphs like AviationKG. Large Language Models (LLMs) have demonstrated efficient performance across various downstream NLP tasks. However, the high computational requirements associated with LLMs have raised concerns. Furthermore, LLMs are typically trained on generic datasets, so their suitability for domain-specific tasks is limited. Knowledge graphs (KGs) are a powerful tool for representing and reasoning over knowledge. They can be used to answer questions, generate text, and make predictions. However, KGs are often incomplete and noisy. This can make it difficult to answer questions that require reasoning over multiple pieces of knowledge. Knowledge infusion is a technique that can be used to improve the performance of DL models on KG-based QA tasks. Knowledge infusion involves injecting knowledge from a KG into a DL model. This can help the model to learn better representations of the knowledge and to reason more effectively over it.

3 Knowledge Graphs

A Knowledge Graph represents triplets $\Omega = < h, r, t >$ in a structured matter. Entities E are represented as nodes, and relations R are defined using a directed edge between the nodes in the head and tail. Knowledge Graphs need not be limited

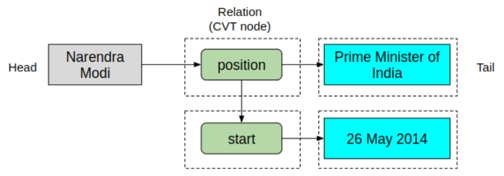


Figure 1: Relation between head and tail showed using a CVT node. The relation "position" has "start" and the start date as additional information

.

to edges between head and tail entities. The edge representing the relation may include more information. Figure 1 shows the head entity as "Narendra Modi" and the tail entity as "Prime Minister of India". The figure also has a relation "position" that mentions the start date. This additional information is represented using nodes to the relations, and the relations are represented as nodes. A node representing relations attaching additional information is known as a Compound Value Type (CVT) node. The figreffig:KnowledgeGraph shows a part of a knowledge graph containing entities as nodes and relations as CVT nodes.

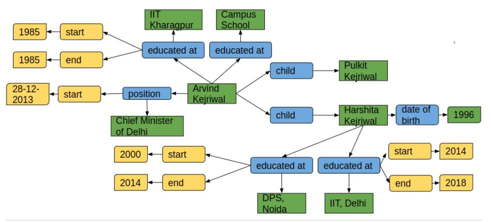


Figure 2: Knowledge graph containing entities and relations. Relations are also shown as nodes instead of edges. Relations also have additional information. Green nodes are entities, blue nodes are relations, and yellow nodes are additional information connecting to the relation's CVT node.

Facts in the Knowledge Graph can be associated with a time stamp. A Knowledge Graph that incorporates timestamps or time intervals is known as Temporal Knowledge Graph. To include time, we add another field to the triplet, i.e., <h,r,t,τ> to get a new tuple. Example: Narendra Modi was born

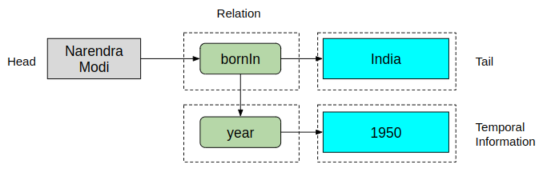


Figure 3: The relation "bornIn" has the date of birth as temporal information

in India in 1950 and is represented as <Narendra Modi, bornIn, India, 1950> and figure 3 shows how it is represented in a Knowledge Graph.

3.1 Ontology

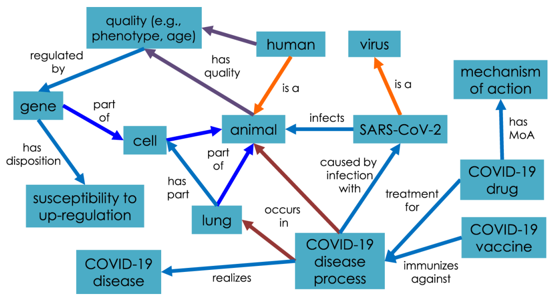


Figure 4: COVID Ontology containing hierarchy 'is-a' and has other relations such as 'infects' and 'treatment for' (He et al., 2020)

An ontology in a knowledge graph (KG) is a formal description of the entities and relationships that exist in a particular domain. It is a way of representing knowledge about a particular domain in a way that is machine-readable.
An ontology typically includes the following:

1. Classes: A class is a collection of entities that share some common properties. For example, the class "Animal" would include all entities that are animals, such as dogs, cats, and horses.

2. Properties: A property is a relationship between two entities. For example, the property "has_color" could be used to relate an entity of class "Animal" to an entity of class "Color".

3. Individuals: Individuals are specific instances of classes. For example, the individual "Spot" is an individual of class "Dog".

Ontologies are used in KGs to provide a common vocabulary for describing the entities and relationships that exist in a particular domain.

3.2 Open Information Extraction Systems

Open information extraction (OIE) is a subfield of natural language processing (NLP) that deals with the automatic extraction of structured information from unstructured text. OIE systems are typically trained on a large corpus of text that has been annotated with the desired information, such as named entities, relations, and events. Once trained, OIE systems can be used to extract information from new text.
OIE systems can be used for a variety of applications, such as question-answering, summarization, and machine learning. For example, an OIE system could be used to extract the following information from the sentence "Barack Obama was born in Honolulu, Hawaii":

1. Named entities: Barack Obama, Honolulu, Hawaii
2. Relations: was born in
3. Events: birth

This information could then be used to answer questions about Barack Obama, such as "Where was Barack Obama born?" or "What is Barack Obama's birthplace?" Stanford OpenIE and OpenIE6 are open-source information extraction (IE) systems that can be used to extract structured information from text. They are both based on the Stanford CoreNLP natural language processing (NLP) toolkit.
Stanford OpenIE was released in 2010 and is based on a rule-based approach to IE. It can extract a variety of information from text, including named entities, relations, and events.
OpenIE6 is an advanced neural Open Information Extraction (OpenIE) system that introduces a novel iterative labeling-based architecture for OpenIE extraction. It improves the performance and capabilities of the OpenIE6 model by incorporating an iterative labeling approach, applying coverage constraints during the training process, and employing the CALMIE model to facilitate the detection of coordination boundaries. These improvements lead to more accurate and comprehensive extractions.

3.3 Entity Linking and Relation Extraction

Entity linking is the task of identifying real-world entities in text and linking them to corresponding entities in a KG. Relation extraction is the task of identifying relationships between entities in text and linking them to corresponding relations in a KG. (Li et al., 2020) identifies entity mentions in a question and scores the entity mention. (Wu et al., 2020) uses a bi-encoder and cross-encoder to disambiguate the entity mentioned. For instance, Jaguar could mean Jaguar vehicle or the animal jaguar. BLINK used Wikidata's description of entities to check semantic similarities with the context in the query.

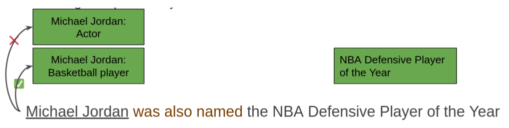

Michael Jordan: Actor
Michael Jordan: Basketball player
NBA Defensive Player of the Year
Michael Jordan was also named the NBA Defensive Player of the Year

Figure 5: Entity Linking: Given a sentence "Michael Jordan was also named the NBA Defensive Player of the Year", determining which Michael Jordan is the text referring to using the context in the sentence

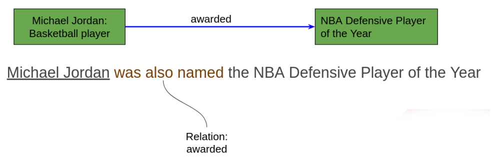

Michael Jordan: Basketball player
awarded
NBA Defensive Player of the Year
Michael Jordan was also named the NBA Defensive Player of the Year
Relation: awarded

Figure 6: Relation Extraction: After identifying entities as "Michael Jordan" and "NBA Defence Player of the Year", extracting the relation between the two using the text that includes both the entities

3.4 Knowledge Graph Construction

Knowledge Graphs can be constructed in a number of ways, but one common approach is first to build an ontology. An ontology is a formal description of the entities and relationships in a domain. It provides a common vocabulary that represents the domain and helps to ensure that the data in the KG is consistent (Weikum et al., 2021).

Once an ontology has been built, triples can be created. Often ontologies for general domain are created by merging multiple existing ontologies and generic KGs such as Cyc (Lenat, 1995), ConceptNet (Speer et al., 2017) and Wordnet (Miller, 1992). Triples can be created manually or automatically using techniques such as open information extraction (refer section 3.2). Manually creating triples can be time-consuming, but it allows for more control over the quality of the data. Automatic triple creation can be faster but may not be as accurate.

3.5 Link Prediction and Knowledge Graph Completion

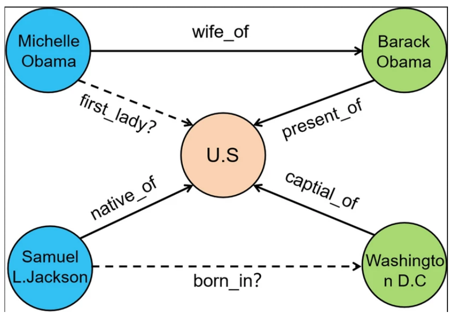


Figure 7: Link Prediction: The solid lines are existing edges in the KG, and dotted lines are missing edges. Link prediction uses a scoring function to predict if a relation should exist between two entities (Huang et al., 2021)

Often Knowledge Graphs are sparse i.e., a knowledge graph may have many facts missing. Such facts can be reasoned using the existing facts. Given an entity and a relation, link prediction is predicting the tail entity. Traversing a knowledge graph is computationally expensive. Knowledge Graph embeddings capture information about their neighbors due to their scoring function. Knowledge Graph Embeddings that capture more properties of relations such as symmetry, asymmetry, anti-symmetry, transitive and n-ary relationships are preferred for the task of completion. Temporal Knowledge Graphs assign timestamps or time-interval to facts. Temporal Knowledge Graph embeddings incorporate time explicitly which is helpful in predicting time and using causal patterns to complete the knowledge graph.

3.6 KG Embeddings

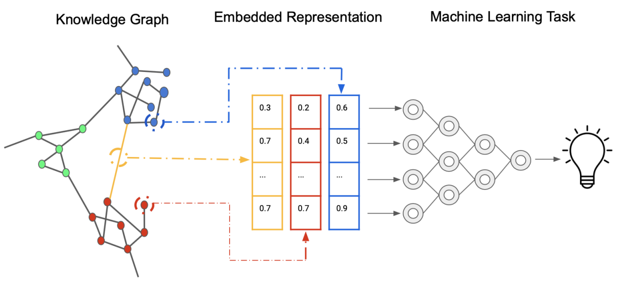


Figure 8: KG Embeddings are trained with the objective of link prediction, Using KGE in a deep learning network allows the DL model to do multihop reasoning over graph (Commons, 2022)

Knowledge graph embeddings are a type of representation learning that maps entities and relations in a knowledge graph to a continuous vector space. This allows us to represent the meaning of entities and relations in a way that can be used by machine learning algorithms. There are two main types of knowledge graph embeddings: translation-based and tensor factorization-based.

1. **Translation-based embeddings** use a scoring function that measures the distance between the embedding of the head entity, the embedding of the relation, and the embedding of the tail entity. The goal is to minimize the distance between the head entity and the tail entity when they are connected by the relation. Translation-based embeddings are relatively simple to train and can be effective for a variety of tasks. However, they can be sensitive to noise in the data and may not be able to capture complex relationships between entities.

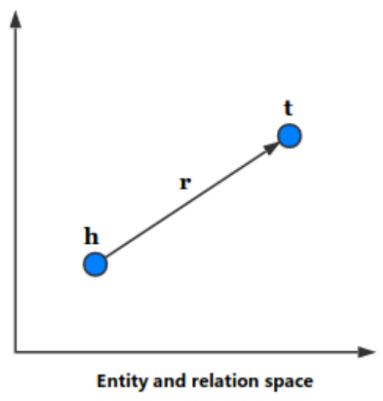


Figure 9: Entities and Relations are represented as vectors, relation is a translation from one entity to another. (Dai et al., 2020)

2. **Tensor factorization-based embeddings** use a tensor factorization model to learn the embedding of entities and relations. The tensor factorization model is trained to predict the existence of a relation between two entities. Tensor factorization-based embeddings are more complex to train than translation-based embeddings, but they can be more effective for capturing complex relationships between entities. However, they can be more computationally expensive to train and may not be as effective for tasks that require real-time inference.

Some popular translation-based embeddings are TransE (Bordes et al., 2013), TransH (Wang et al.,

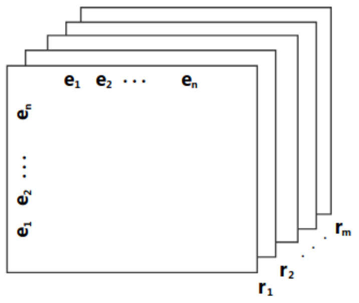


Figure 10: Knowledge Graph represented as tensor, where entities are two of the dimensions and the third dimension is relation (Dai et al., 2020)

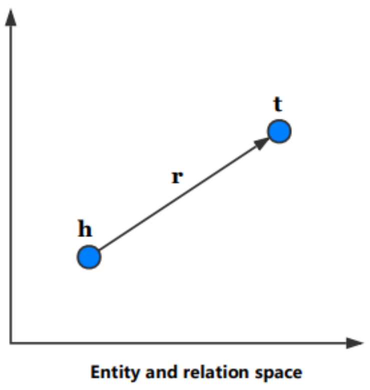


Figure 11: TransE: Translation over hyperplane, here **r** is translation from **h** to **t**. (Dai et al., 2020)

2014), TransR (Lin et al., 2015), and HyTE (Wang et al., 2014). Some popular tensor factorization-based embeddings are RESCAL (Nickel et al., 2011), DistMult (Yang et al., 2015), and ComplEx (Trouillon et al., 2016). Knowledge graph embeddings have been shown to be effective for a variety of tasks, including link prediction and question answering.

3.6.1 Translation Based Models

Translation-based Knowledge Graph Embedding models represent relation as a translation from a head entity to a tail entity in an embedding space. The proposed translation-based models vary in the space these elements are projected. The following are some translation-based models:

1. TransE: TransE (Bordes et al., 2013)is the first Knowledge Graph Embedding method which translates head by the relation to reach the tail entity keeping both head and tail entities in the same dimension i.e. k=d. The figure 11 shows the translation from head to tail using a relation.
   The scoring function of TransE is f$_r$(h,t) = $||\mathbf{h}+\mathbf{r}+\mathbf{t}||_{l_1/l_2}$. The model complexity of TransE is $\mathcal{O}(N_e d + N_r k)(d = k)$. TransE fails to model the one-to-many, many-to-one and many-to-many relationship between entities. It can't model symmetric and reflexive relations.

2. TransH: TransH (Wang et al., 2014) projects the head and tail entities onto a relation specific hyperplane and then uses translation on the hyperplane to translate the projected head to obtain the projected tail entity. Since multiple entities can be projected to the same point on the hyperplane, it can model one-to-many, many-to-one, and many-to-many relations.
   The scoring function of TransH is f$_r$(h,t) = $||(\mathbf{h}-\mathbf{w}_r^T\mathbf{h}\mathbf{w}_r)+\mathbf{d}_r-(\mathbf{t}-\mathbf{w}_r^T\mathbf{t}\mathbf{w}_r)||_2^2$. The model complexity of TransH is $\mathcal{O}(N_e d + N_r k)(d = k)$. The ability to model n-ary relationship is limited due to the projection onto a hyperplane in the same space as entities.

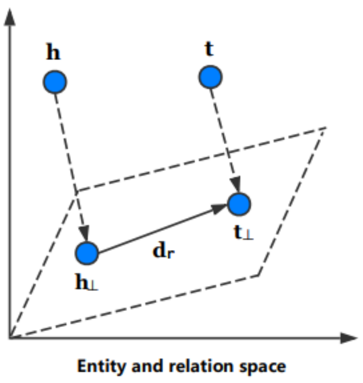


Figure 12: TransH: Entities are first projected onto hyperplaned then translated over it. (Dai et al., 2020)

3. TransR: TransR (Lin et al., 2015) projects the head and tail entities from a k-dimensional space to a d-dimensional space. Projection into a seperate space allows TransR to model more n-ary relations. The projection is performed using a relation-specific projection matrix.
   The scoring function of TransR is f$_r$(h,t) = $||(\mathbf{M}_r\mathbf{h}+\mathbf{r}-\mathbf{M}_r\mathbf{t}||_2^2$. The model complexity of TransR is $\mathcal{O}(N_e d + N_r dk)$. TransR increased the complexity of parameters due to the projection matrix $\mathbf{M}_r$.

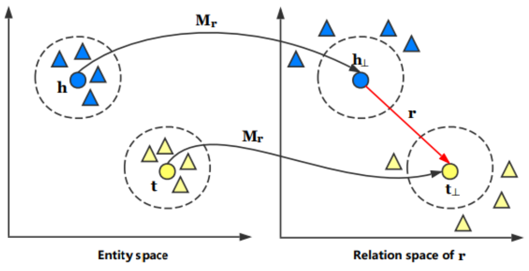


Figure 13: TransR: Projection matrices are used to project entities to relation space. (Dai et al., 2020)

4. TransD: TransD (Ji et al., 2015) projects the head and tail entities from a k-dimensional space to a d-dimensional space. Unlike TransR, TransD uses seperate projection matrix for head and tail entity. TransD reduces the number of parameters by obtaining the projection matrices using vector multiplication.
   The projection matrix for head entity is obtained using vectors $\mathbf{r}_p$ $\mathbf{h}_p^T$ for tail using vectors $\mathbf{r}_p$ and $\mathbf{t}_p^T$. The scoring function of TransD is f$_r$(h,t) = $||(\mathbf{r}_p\ \mathbf{h}_p^T+\mathbf{I})\mathbf{h} + \mathrm{r} - (\mathbf{r}_p\ \mathbf{t}_p^T+\mathbf{I})\mathbf{t}\ ||_2^2$. The model complexity of TransD is $\mathcal{O}(N_e d + N_r k)$.

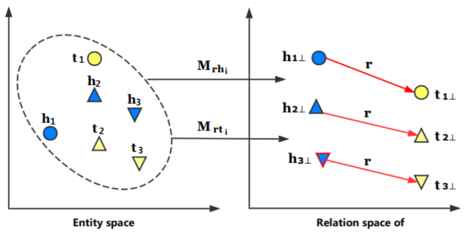


Figure 14: TransD: Sparse projection matrices are used to project entities to relation space.The Sparse projection matrices can be represented as multiplication of two vectors forming a projection matrix. (Dai et al., 2020)

3.6.2 Tensor Factorization Based Models

Tensor factorization-based methods represent entities and relation between them as a tensor, as shown in figure 15. The tensor representing the knowledge graph is a three-dimensional binary matrix X ϵ R^{n,n,m} where n is the number of entities and m is the number of relations. Tensor factorization-based methods decompose the KG tensor into a multiplication of factors as entities and relation. The following are some Tensor-factorization based methods:

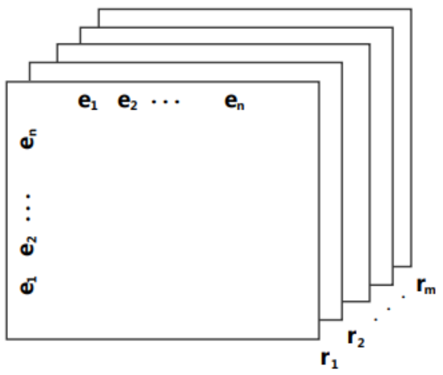


Figure 15: Entities and relation represented as a three-dimensional tensor (Dai et al., 2020)

1. RESCAL: RESCAL expresses the tensor into representation of head, relation and tail.
   The scoring function of RESCAL is f$_r$(h,t) = $\mathbf{h}^T\mathbf{M}_r\mathbf{t}$. The model complexity of RESCAL is $\mathcal{O}(N_e d + N_r k^2)(d = k)$. The model complexity of RESCAL is quadratic in k.

2. DistMult: DistMult restricts the relation matrix to a diagonal matrix, hence reducing the number of parameters required by a relation. It allows symmetric relations between head and tail entities.
   The scoring function of DistMult is f$_r$(h,t) = $\mathbf{h}^T$diag($\mathbf{r}$)$\mathbf{t}$. The model complexity of DistMult is $\mathcal{O}(N_e d + N_r k)(d = k)$.

3. HolE: HolE simplifies the tensor product by introducing circular correlation. The circular correlation operation is indicated using $\star$. For two entities, it can be calculated as $[\mathbf{h}\star\mathbf{t}]_k = \sum_{i=0}^{d-1} h_i t_{(k+i) mod d}$. The operation is asymmetric which allows the model to represent asymmetric relations. Also, the complexity can be further improved using fast fourier transform(FFT) using $h \star t = \mathcal{F}^{-1}(\overline{\mathcal{F}(\mathbf{h})} \odot \mathcal{F}(\mathbf{t}))$.
   The scoring function of HolE is f$_r$(h,t) = $\mathbf{r}^T(\mathbf{h}\star\mathbf{t})$. The model complexity of HolE is $\mathcal{O}(N_e d + N_r k)(d = k)$.

4. ComplEx: ComplEx uses embeddings from complex space i.e. C^d. Because of the complex space, along with symmetric, asymmetric relations can be modeled.
   The scoring function of ComplEx is f$_r$(h,t) = Re($\mathbf{h}^T$diag($\mathbf{r}$)$\bar{\mathbf{t}}$) where $\bar{\mathbf{t}}$ is complex conjugate of t and Re() returns the real part of a complex

Datasets	WN18				FB15K			
Metric	Mean Rank		HITS@10(%)		Mean Rank		Hits@10(%)	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE	263	251	75.4	89.2	243	125	34.9	47.1
TransH	401	388	73.0	82.3	212	87	45.7	64.4
TransR	238	225	79.8	92.0	198	77	48.2	68.7
TransD	224	212	79.6	92.2	194	91	53.4	77.3
RESCAL	1180	1163	37.2	52.8	828	683	28.4	44.1
DistMult -	-	-	-	94.2	-	-	-	58.5
HOLE	-	-	-	94.9	-	-	-	73.9
ComplEx	-	-	-	94.7	-	-	-	84.0

Table 1: Evaluation of Knowledge Graph Embedding Models on WN18 and FB15K datasets (Dai et al., 2020)

value. The model complexity of ComplEx is $\mathcal{O}(N_e d + N_r k)(d = k)$.

4 Deep Learning for NLP

Deep learning in natural language processing (NLP) is a subfield of artificial intelligence that focuses on using neural networks to process and understand human language. It leverages deep neural networks, which are composed of multiple layers of interconnected artificial neurons, to automatically learn intricate patterns and representations from textual data. The combination of linguistics and deep learning in NLP has revolutionized various NLP tasks, including question-answering, machine translation, sentiment analysis, text summarization, and more. Deep learning for NLP uses word embeddings, RNNs, and attention mechanisms to capture meaning and dependencies in text. Transformers are a newer architecture that can parallelize computation and capture long-range dependencies more effectively. Pretraining and transfer learning improve performance on downstream tasks with smaller datasets.

4.1 Word Embeddings

Word embeddings are dense vector representations of words that capture their semantic and syntactic relationships. They are a key component of many deep learning models for natural language processing (NLP) tasks, such as machine translation, text classification, and question answering. There are two main approaches to learning word embeddings: continuous bag-of-words (CBOW) and skip-gram. CBOW predicts the target word given its context words, while skip-gram predicts the context words given the target word. Both CBOW and skip-gram are neural network models that are trained on large corpora of text. The input to the model is a sequence of words, and the output is a vector representation of the target word. The model is trained to minimize the error between the predicted vector representation and the ground truth vector representation. Word embeddings have been shown to be effective because of their ability to capture the semantic and syntactic relationships between words, which allows models to learn more complex representations of language.

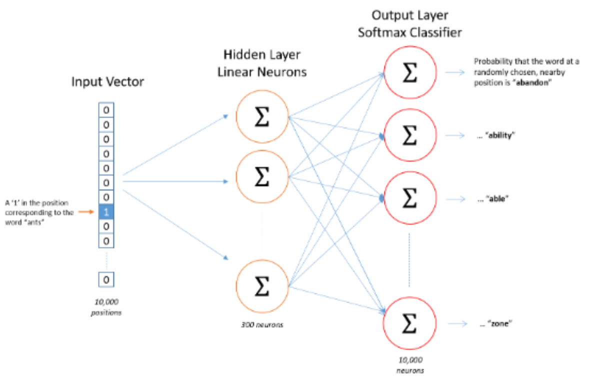


Figure 16: Skipgram: The input is a one-hot vector, here the vector is set at the position of ‘ant’ and 0 elsewhere. The output is one hot vector for each of its contexts. (Bhattacharyya, 2022)

4.2 Recurrent Neural Networks and Transformers

Recurrent Neural Networks (RNNs) are a type of neural network that are designed to process sequential data by utilizing the concept of feedback loops. They are particularly effective in tasks that involve sequential or temporal dependencies.

The key idea of RNNs is their ability is that they maintain an internal state or memory which captures previous history previous inputs in the se-

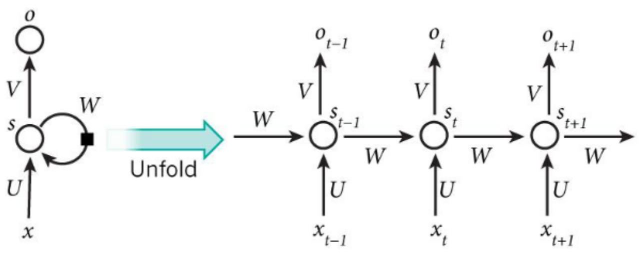


Figure 17: The basic architecture of a recurrent neural network (RNN). (Bhattacharyya, 2022)

quence. This internal state is passed along through the network, allowing it to retain information about the context and history of the sequence. This is achieved by introducing recurrent connections, where the output of a hidden layer at a given time step is fed back as an input to the same layer at the next time step.
As shown in figure 17, at each time step, an RNN takes an input vector and combines it with the internal state from the previous time step to produce an output and update the internal state. This process is repeated for each element in the sequence, allowing the network to capture the sequential dependencies.
An RNN can be described by the following equations:

$$\text{Hidden state update:} s_t = f(W * s_{t-1} + U * x_t + b_h) \tag{1}$$

$$\text{Output computation:} y_t = f(V * s_t + b_o) \tag{2}$$

Here, s_t represents the hidden state at time step t, x_t is the input at time step t, y_t is the output at time step t, and $f()$ is an activation function. W, U, V, b_h, and b_o are weight matrices and bias vectors that are learned during the training process.
One important variant of RNNs is the **Long Short-Term Memory (LSTM) network**. LSTMs address the issue of vanishing or exploding gradients that can occur in traditional RNNs by incorporating memory cells and gating mechanisms. These memory cells allow LSTMs to selectively remember or forget information over long sequences, making them more effective in capturing long-term dependencies.
In natural language processing, RNNs are used for tasks such as language modeling, machine translation, sentiment analysis, and question answering.

Unlike traditional recurrent neural networks

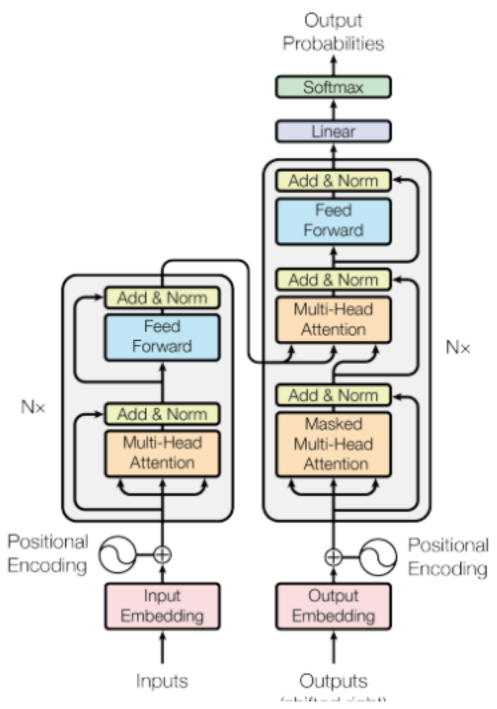


Figure 18: Architecture of a Transformer model (Vaswani et al., 2017)

(RNNs) that process sequential data, **transformers** (Vaswani et al., 2017) are based on a self-attention mechanism that enables parallel processing of input sequences, making them highly efficient and effective. Self-attention is a mechanism that allows the model to focus on different parts of the input sequence when encoding or decoding. It assigns weights to each input position based on its relevance to other positions in the sequence, capturing dependencies and relationships. RNNs often suffer from the vanishing gradient problem, where gradients diminish exponentially as they propagate through multiple time steps. This limitation hampers the ability of RNNs to capture long-range dependencies effectively. Transformers alleviate this issue by employing residual connections and layer normalization.

4.3 Language Models

A language model is a statistical method that predicts the next word in a sequence. Language models are used in a variety of natural language processing (NLP) tasks, such as question-answering, machine translation, and text generation.

4.3.1 Pretraining and Finetuning

In pretraining, a language model is trained on a large corpus of text with a general objective, such as predicting the next word in a sequence. This helps the model learn the statistical relationships between words and phrases. In finetuning, the language model is trained on a smaller dataset of text with a specific task, such as question answering or machine translation. This helps the model learn the specific features of the task and improve its performance.

4.3.2 Architectures of Language Models

There are different architectures for language models, each serving specific purposes in natural language processing tasks.

1. Encoder-Only Models:
   (a) Encoder-only models focus on encoding input sequences into meaningful representations.
   (b) These models capture contextual information from the input and create fixed-length representations, often referred to as embeddings.
   (c) Examples of encoder-only models include XLNet (Yang et al., 2019), RoBERTa (Liu et al., 2019) and BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019).
   (d) Encoder-only models are widely used for tasks such as text classification, named entity recognition, and sentiment analysis, where the focus is on understanding the input.
2. Decoder-Only Models:
   (a) Decoder-only models concentrate on generating output sequences based on given input or context.
   (b) These models take an initial input or representation and generate sequences step by step, often using autoregressive approaches.
   (c) Decoder-only models are commonly used in tasks like machine translation, text generation, and image captioning.
   (d) Prominent examples include the decoder part of sequence-to-sequence models and generative language models like GPT (Generative Pre-trained Transformer) and BLOOM
3. Encoder-Decoder Models:
   (a) Encoder-decoder models combine both encoding and decoding components to handle tasks involving input and output sequences.
   (b) These models encode the input sequence into a fixed-length representation and then use the decoder to generate the output sequence.
   (c) Encoder-decoder models are widely used in machine translation, question-answering, and text summarization.
   (d) Popular architectures include the sequence-to-sequence model with attention mechanisms (e.g. T5) and the transformer-based model for machine translation (e.g., Transformer for Neural Machine Translation - NMT).

5 Information Retrieval

Information retrieval plays a vital role in managing and extracting relevant information from vast amounts of textual data. We delve into different types of neural ranking models, including bi-encoder and cross-encoder models, and discuss their architectures and training strategies.

5.1 Early and Late Interaction Models

Early and late interaction models are two different approaches used in information retrieval systems for ranking passages based on a given query.

5.1.1 Early Interaction Models

Early interaction models process the query and passages together from the beginning of the ranking process. In these models, the query and passage representations are combined or fused early on, usually before any scoring or ranking is performed. One common approach in early interaction models is to concatenate the query and passage representations into a single vector or matrix. This concatenated representation is then fed into a scoring function such as a model like BERT to determine the relevance between the query and passage. The scores obtained from the model are used to rank the passages.

Early interaction models have the advantage of capturing the interaction between the query and passage from the start. Early interaction methods perform well but since for k passages, the similarity is to be computed k times through a model, it is very slow making it difficult to scale over large number of documents.

5.1.2 Late Interaction Models

Late interaction models, on the other hand, process the query and passages separately and perform interaction at a later stage, typically after obtaining individual representations for the query and passages.

In late interaction models, the query and passage representations are computed independently using pre-trained language models such as BERT or GPT. These representations capture the contextual information of the query and passages. After obtaining the representations, a scoring function is applied to measure the relevance or similarity between the query and each passage separately.

The scores obtained for each passage are then used for ranking. The late interaction allows for more flexibility in capturing the interaction patterns between the query and passages. It allow the model to precompute the embeddings for each passage allowing it to scale. The scoring function is usually a dot product or cosine similarity which are faster to compute.

5.2 BM25 and Elastic Search

BM25 is a ranking function used to determine the relevance of documents to a query, while Elasticsearch is a scalable search and analytics engine that utilizes BM25 and other algorithms for efficient and powerful search capabilities.

5.2.1 BM25

BM25 (Robertson and Zaragoza, 2009), or the Okapi BM25 algorithm, is a statistical method for ranking documents in a search engine results page (SERP) based on their relevance to a user's query. It is a term weighting algorithm, which means that it assigns different weights to different terms in a document, depending on how frequently they occur in the document and in the corpus of documents as a whole.

BM25 works by first calculating a document frequency (DF) for each term in the document. The DF is the number of documents in the corpus that contain the term. BM25 then calculates a term frequency (TF) for each term in the document. The TF is the number of times the term occurs in the document.

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)} \tag{3}$$

where D is the document, Q is the query, n is the number of terms in the query, q_i is the ith term in the query, $f(q_i, D)$ is the frequency of q_i in D, $|D|$ is the length of D in words, $avgdl$ is the average document length in the text collection, k_1 and b are free parameters and $IDF(q_i)$ is the IDF weight of q_i. Typically $k_1 \in [1.2, 2.0]$ and b = 0.75.

The IDF weight is calculated as follows:

$$IDF(q_i) = \ln\left(\frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1\right) \tag{4}$$

where N is the total number of documents in the collection and n(q_i) is the number of documents containing q_i

5.2.2 Elastic Search

Elasticsearch is an open-source search engine built on top of Apache Lucene, and it provides a powerful and flexible way to store, search, and analyze data.

Elasticsearch is often used for a variety of tasks, including:

1. Log analytics: Elasticsearch can be used to collect, store, and analyze log data from applications and servers. This can be used to troubleshoot problems, identify performance bottlenecks, and monitor system health.

2. Full-text search: Elasticsearch can be used to index and search large amounts of text data. This can be used to power search features on websites, applications, and internal systems.

3. Real-time analytics: Elasticsearch can be used to analyze data in real time. This can be used to track changes in data over time, identify trends, and detect anomalies.

Elasticsearch is a distributed system, which means that it can be scaled horizontally to handle large amounts of data. Elasticsearch nodes are organized into clusters, and each node can store and process data.

Elasticsearch uses a document-oriented data model. This means that data is stored in documents, which

are JSON objects. Documents are indexed into shards, which are logical divisions of the data. Shards can be distributed across multiple nodes in the cluster.
When a user performs a search, Elasticsearch uses a query language called Elasticsearch Query DSL to find the documents that match the search criteria. The search results are then ranked using a scoring algorithm, such as BM25.

5.3 Bi-Encoder and Cross-Encoder

Bi-encoders and cross-encoders are both used in information retrieval (IR) to find relevant passages for a given query. In IR, a query is a short piece of text that represents the user's information needs. A passage is a longer piece of text that contains information that may be relevant to the user's information needs.

5.3.1 Bi-Encoder

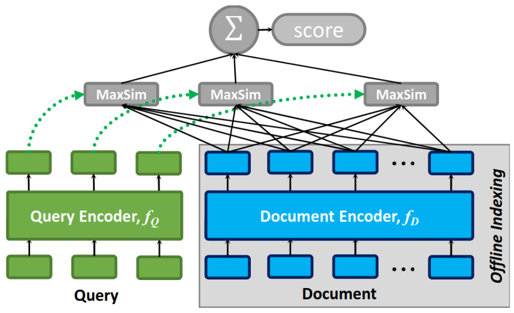


Figure 19: Bi-encoder from (Khattab and Zaharia, 2020) where representations of query and passage are used to check similarity

A bi-encoder is a model that independently encodes two pieces of text, such as a question and a passage, into a vector representation. The vector representations are then used to calculate a similarity score, which is used to determine whether the question and passage are related.
Bi-encoders are relatively simple to train and can be effective for a variety of NLP tasks, such as question answering and natural language inference. However, bi-encoders can be limited in their ability to capture long-range dependencies between words in a text.

5.3.2 Cross-Encoder

A cross-encoder is a model that jointly encodes two pieces of text into a single vector representation. The vector representation is then used to perform a variety of NLP tasks, such as question answering, natural language inference, and sentiment analysis.

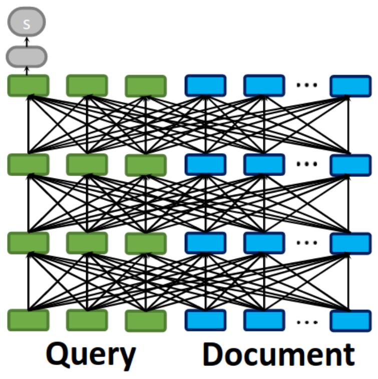


Figure 20: Cross-encoder from (Khattab and Zaharia, 2020) where a BERT-like model is used to obtain similarity by passing query and document as input to the encoder and allow early interaction through attention.

Cross-encoders are more complex than bi-encoders, but they can be more effective for tasks that require the model to understand the relationship between two pieces of text. Cross-encoders can also capture long-range dependencies between words in a text, which can be helpful for tasks such as question answering.

5.4 Retrieve and Rerank

Retrieve and rerank is a mechanism used in information retrieval (IR) to improve the accuracy of the results. In retrieve, a bi-encoder is used to identify a set of relevant documents for a given query. In rerank, a cross-encoder is used to rank the documents in the set according to their relevance to the query. The retrieve and rerank mechanism can be helpful because it allows the IR system to take advantage of the strengths of both bi-encoders and cross-encoders. Bi-encoders are fast and easy to train, while cross-encoders are more accurate. By using both bi-encoders and cross-encoders, the IR system can identify a large set of relevant documents quickly and then rank the documents first few documents in the set according to their relevance to the query. The workflow of retrieve-rerank is shown in figure 21. For example, a user enters the query "What is the capital of France?" into a search engine or QA system. The system uses a bi-encoder to identify a set of relevant documents for the query. The documents in the set are then ranked by a cross-encoder according to their relevance to the query. The document with the highest ranking is the document that is most likely to contain the answer to the query.
In this example, the bi-encoder would identify a set

of documents that contain the words "France" and "capital." The cross-encoder would then rank the documents in the set according to their similarity to the query. The document with the highest ranking would be the document that is most likely to contain the answer to the query. In this case, the document with the highest ranking would be the document stating that France's capital is Paris.

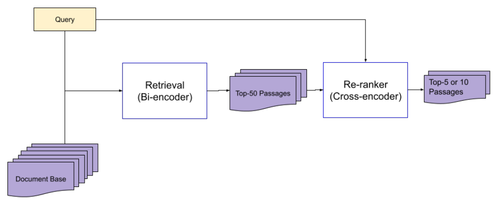


Figure 21: The retrieve-rerank retriever uses a bi-encoder and then a cross-encoder to retrieve relevant documents for a query. The bi-encoder uses cosine similarity to fetch the top 20 documents, and then these 50 passages are reranked using a cross-encoder. The top-5 documents ranked by cross-encoder are used for question answering

5.5 ColBERT

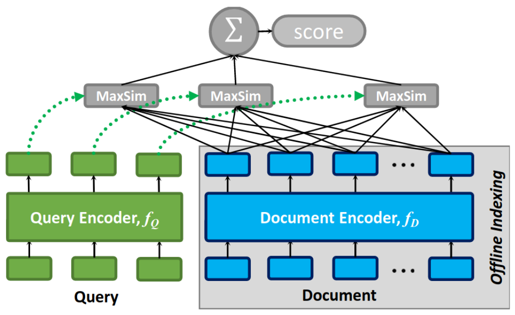


Figure 22: ColBERT matching relevant text spans between query tokens and document tokens to obtain a similarity score (Khattab and Zaharia, 2020)

ColBERT (Khattab and Zaharia, 2020) stands for "Contextualized Late Interaction over BERT" and is specifically designed for passage ranking in information retrieval. It aims to improve the effectiveness of retrieving relevant passages from large collections of documents given a user query.
The ColBERT model builds upon the BERT (Bidirectional Encoder Representations from Transformers) architecture, a widely used language representation model. BERT is trained on large amounts of text data to generate contextualized word embeddings that capture the semantic meaning of words based on their context.

ColBERT takes the BERT architecture and modifies it to optimize for the passage ranking task. It processes the query and passages separately to obtain their respective embeddings of tokens. As shown in figure 22, it then applies a late interaction mechanism, where the embeddings of the query and passage are combined to compute relevance scores. The spans of the query are matched with the spans of the document. The similarity between spans is summed up to obtain a similarity score. This allows ColBERT to effectively capture the interaction between the query and passage representations.

6 Question Answering

Question-Answering (QA) is a natural language processing task that seeks to provide the answer to a given question. In this section, we will discuss open-book and closed-book setting for QA. QA models are often trained on general domain data and perform poorly in specialized domains. We will discuss the challenges in domain-specific QA. Finally, we discuss the work on knowledge-infused QA.

6.1 Open Book and Closed Book QA

Open book QA allows users to access any information they want while answering the question. This means that they can use any resources they have available, such as books, articles, or the internet. Closed book QA does not allow the user to access any information while answering the question. This means that they must rely on their own knowledge and understanding of the topic to answer the question.
Open-book QA is typically easier than closed-book QA because the user has access to more information. However, it can also be more time-consuming because the user has to sift through all of the information to find the answer. Closed-book QA is typically more challenging than open-book QA because the user has to rely on their own knowledge and understanding of the topic. However, it can also be more efficient because the user does not have to waste time looking for information.

6.2 Challenges in Domain Specific QA

The following are the challenges that arise with domain-specific QA:

1. Data scarcity: There is often less data available for training a domain-specific QA sys-

tem than for a general-purpose QA system. This is because domain-specific data is often more specialized and difficult to collect. For instance, in-depth investigation about accidents or patients which might not be available on Wikipedia or the news, which are primary sources of data collection for pretraining data. When less data is used to train models, the model struggles to learn the domain knowledge due to underfitting.

2. Domain knowledge: Domain-specific QA systems need to have a good understanding of the domain in order to answer questions accurately. This can be difficult to achieve, especially for domains that are complex or have a lot of jargon. For. example Bear Market and Limit Order are words from financial domain that are not used in day-to-day conversations and writing.

3. Closed-book QA: In real-world QA, the system is not allowed to access external resources, such as the Internet, to answer questions. This makes it more difficult for the system to answer questions that require outside knowledge.

4. Domain knowledge graphs (KGs): Domain KGs can be helpful for domain-specific QA systems, but they are often incomplete or inaccurate and rare for specialized domains. This can lead to the system providing incorrect or incomplete answers.

5. Data distribution shift: The distribution of data in the real world may be different from the distribution of data used to train the model, which can lead to the model performing poorly on real-world data.

6.3 Knowledge Graph and Deep Learning based Question Answering Systems

Traditional approaches to querying KGs in Natural Language (NL) involved rule-based and pattern-based systems, along with semantic parsing techniques that converted NL questions into symbolic queries over the KG. However, recent advancements have shifted towards the use of sequence-to-sequence (seq2seq) architectures and pre-trained models, leveraging the power of neural networks. The integration of KGs and deep learning (DL) has attracted significant research attention in the field of natural language query retrieval, as it addresses the challenge of converting NL into graph query language. One successful approach involves combining structured knowledge with DL by converting the knowledge into natural language text. ERNIE 3.0 (Sun et al., 2021), for example, trains a knowledge-enhanced model on a corpus that combines triples (KG statements) and their corresponding sentences. During training, random masking is applied to either the relation in a triple or words in a sentence. Other methods like QA-GNN (Yasunaga et al., 2021) and GreaseLM (Zhang et al., 2022) employ knowledge infusion techniques, propagating information through a graph to capture dependencies and relationships among entities.
The synergy between KGs and DL can be categorized into two groups. First, utilizing KGs during inference, as shown in PullNet (Sun et al., 2019). Second, infusing knowledge into model weights during pre-training, as explored in approaches such as K-BERT(Liu et al., 2020), KGT5(Saxena et al., 2022), and SKILL (Moiseev et al., 2022).

7 Datasets

In this section, we will discuss important datasets for Question Answering, Aviation documents, and benchmark Knowledge Graphs used for link prediction and knowledge infusion.

7.1 Aviation Documents

The National Transportation Safety Board (NTSB)[1] and Airworthiness directives (ADs) [2] are two important safety organizations that work together to improve aviation safety. The NTSB investigates accidents and identifies unsafe conditions, while the FAA issues ADs to address these unsafe conditions. The ADREP taxonomy is a list of terms and definitions used to describe and classify aviation accidents and incidents. It is used by aviation safety professionals to share information and develop effective safety measures.

7.1.1 National Transport Safety Board Reports

The National Transportation Safety Board (NTSB) is an independent federal agency that investigates transportation accidents, including airplane crashes. The NTSB's goal is to prevent future accidents by

[1] https://www.ntsb.gov/investigations/Pages/Investigations.aspx
[2] https://www.faa.gov/regulations_policies/airworthiness_directives

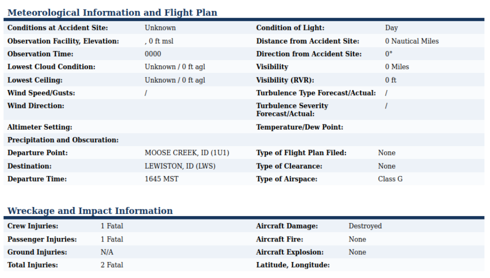
identifying the causes of accidents and making recommendations for safety improvements. NTSB aviation reports are typically divided into four sections:

1. Summary: This section provides a brief overview of the accident, including the date, time, location, and number of fatalities.

2. Investigation: This section provides a detailed description of the NTSB's investigation, including the interviews of witnesses, the examination of the wreckage, and the analysis of the data.

3. Analysis: This section analyzes the accident and identifies the probable cause.

4. Findings and Recommendations: This section makes safety recommendations to prevent similar accidents from happening in the future.



Meteorological Information and Flight Plan

Conditions at Accident Site:	Unknown	**Condition of Light:**	Day
Observation Facility, Elevation:	, 0 ft msl	**Distance from Accident Site:**	0 Nautical Miles
Observation Time:	0000	**Direction from Accident Site:**	0°
Lowest Cloud Condition:	Unknown / 0 ft agl	**Visibility**	0 Miles
Lowest Ceiling:	Unknown / 0 ft agl	**Visibility (RVR):**	0 ft
Wind Speed/Gusts:	/	**Turbulence Type Forecast/Actual:**	/
Wind Direction:		**Turbulence Severity Forecast/Actual:**	/
Altimeter Setting:		**Temperature/Dew Point:**	
Precipitation and Obscuration:			
Departure Point:	MOOSE CREEK, ID (1U1)	**Type of Flight Plan Filed:**	None
Destination:	LEWISTON, ID (LWS)	**Type of Clearance:**	None
Departure Time:	1645 MST	**Type of Airspace:**	Class G

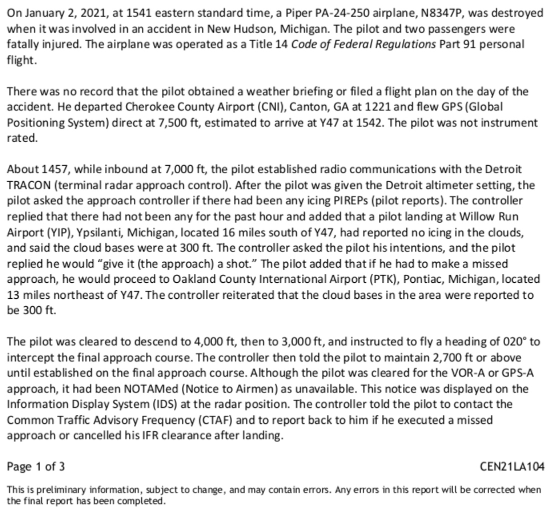
Wreckage and Impact Information

Crew Injuries:	1 Fatal	**Aircraft Damage:**	Destroyed
Passenger Injuries:	1 Fatal	**Aircraft Fire:**	None
Ground Injuries:	N/A	**Aircraft Explosion:**	None
Total Injuries:	2 Fatal	**Latitude, Longitude:**	

Figure 23: Structured part of NTSB report of accident number SEA87LA080. It contains meteorological information and flight plan

NTSB Reports has some information in structured tabular format while rest in unstructured text format.

1. **Structured information** is organized in a consistent format and extraction of specific details become easy to extract. NTSB reports contains one or more tables that contains information on findings, history of flight, pilot information, aircraft and owner/operator information, meteorological information and flight plan, airport information, wreckage and impact information and administrative information. Each table goes into further details. Many of these details are present as unstructured format in the report as well. Structured information is very helpful to create a database or a knowledge base to store information about the accident. Figure 23 shows



On January 2, 2021, at 1541 eastern standard time, a Piper PA-24-250 airplane, N8347P, was destroyed when it was involved in an accident in New Hudson, Michigan. The pilot and two passengers were fatally injured. The airplane was operated as a Title 14 *Code of Federal Regulations* Part 91 personal flight.

There was no record that the pilot obtained a weather briefing or filed a flight plan on the day of the accident. He departed Cherokee County Airport (CNI), Canton, GA at 1221 and flew GPS (Global Positioning System) direct at 7,500 ft, estimated to arrive at Y47 at 1542. The pilot was not instrument rated.

About 1457, while inbound at 7,000 ft, the pilot established radio communications with the Detroit TRACON (terminal radar approach control). After the pilot was given the Detroit altimeter setting, the pilot asked the approach controller if there had been any icing PIREPs (pilot reports). The controller replied that there had not been any for the past hour and added that a pilot landing at Willow Run Airport (YIP), Ypsilanti, Michigan, located 16 miles south of Y47, had reported no icing in the clouds, and said the cloud bases were at 300 ft. The controller asked the pilot his intentions, and the pilot replied he would "give it (the approach) a shot." The pilot added that if he had to make a missed approach, he would proceed to Oakland County International Airport (PTK), Pontiac, Michigan, located 13 miles northeast of Y47. The controller reiterated that the cloud bases in the area were reported to be 300 ft.

The pilot was cleared to descend to 4,000 ft, then to 3,000 ft, and instructed to fly a heading of 020° to intercept the final approach course. The controller then told the pilot to maintain 2,700 ft or above until established on the final approach course. Although the pilot was cleared for the VOR-A or GPS-A approach, it had been NOTAMed (Notice to Airmen) as unavailable. This notice was displayed on the Information Display System (IDS) at the radar position. The controller told the pilot to contact the Common Traffic Advisory Frequency (CTAF) and to report back to him if he executed a missed approach or cancelled his IFR clearance after landing.

Page 1 of 3 CEN21LA104

This is preliminary information, subject to change, and may contain errors. Any errors in this report will be corrected when the final report has been completed.

Figure 24: Unstructured part of NTSB report of accident number CEN21LA104. It contains the events that occurred before the accidents.

.

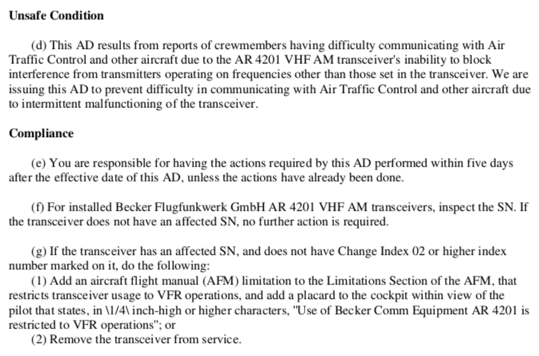
the meteorological information and flight plan, and wreckage and impact information. The meteorological information contains the data about departure point and weather and visibility conditions. Wreckage information contains details about the injuries and aircraft damage.

2. **Unstructured information** is not organized in a consistent format. The investigation, analysis and findings are explained in detail in the form of sentences. This can include statements by witnesses, details of the sequence of events before the accident and that lead to accident, details of findings by inspections and some preventive measures for new type of accidents. These details are specific information to the accident and such information cannot be generalized to a structure across all formats. Figure 24 explains the events that took place before the accident occurred.

7.1.2 Airworthiness Directives

An Airworthiness Directive (AD) is a document issued by the Federal Aviation Administration (FAA) that requires aircraft owners and operators to take action to correct an unsafe condition in an aircraft. ADs can be issued for a variety of reasons, including design flaws, manufacturing defects, maintenance errors, or in-service failures. ADs are issued to help prevent accidents by addressing unsafe conditions in aircraft. By requiring operators to take action to correct these conditions, ADs help to ensure that the aviation industry is as safe as possible. ADs are not always easy to comply with. Sometimes, the corrective action required by an AD can be complex or expensive. Therefore typically ADs mention the cost of compliance in their reports.

ADs can be emergency or regular based on the



Unsafe Condition

(d) This AD results from reports of crewmembers having difficulty communicating with Air Traffic Control and other aircraft due to the AR 4201 VHF AM transceiver's inability to block interference from transmitters operating on frequencies other than those set in the transceiver. We are issuing this AD to prevent difficulty in communicating with Air Traffic Control and other aircraft due to intermittent malfunctioning of the transceiver.

Compliance

(e) You are responsible for having the actions required by this AD performed within five days after the effective date of this AD, unless the actions have already been done.

(f) For installed Becker Flugfunkwerk GmbH AR 4201 VHF AM transceivers, inspect the SN. If the transceiver does not have an affected SN, no further action is required.

(g) If the transceiver has an affected SN, and does not have Change Index 02 or higher index number marked on it, do the following:
(1) Add an aircraft flight manual (AFM) limitation to the Limitations Section of the AFM, that restricts transceiver usage to VFR operations, and add a placard to the cockpit within view of the pilot that states, in \1/4\ inch-high or higher characters, "Use of Becker Comm Equipment AR 4201 is restricted to VFR operations"; or
(2) Remove the transceiver from service.

Figure 25: Snapshot of Docket No. 2003-NE-68-AD, the figure shows unsafe condition and compliance. The information is unstructured. The format of the information differs from report to report.

.

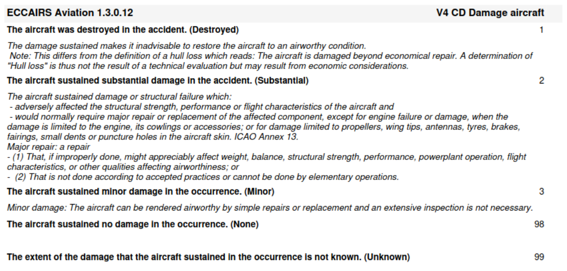
urgency and time of compliance. Emergency ADs (EADs)are issued when there is an immediate safety hazard. EADs must be complied with immediately, or the aircraft may not be allowed to fly. Regular ADs are issued when there is a less urgent safety hazard. Regular ADs must be complied with within a specified period of time, which is usually 12 months. ADs are legally enforceable regulations. This means that aircraft owners and operators who do not comply with an AD may be subject to civil penalties or criminal charges. Each AD is associated with a unique docket number. Figure 25 shows a snapshot of Docket SEA87LA080. It contains meteorological information and flight plan No. 2003-NE-68-AD which contains the unsafe condition, the part which caused the condition and actions taken to resolve the unsafe condition. ADs have information in unstructured format and can be difficult to create a Knowledge Base from it.

Unlike NTSB reports ADs are not the same as accident reports. Accident reports are typically issued after an accident has occurred, and they are designed to determine the cause of the accident. ADs, on the other hand, are issued to prevent accidents from happening in the first place.

7.1.3 ADREP Taxonomy

The ADREP taxonomy is a set of terms used by the International Civil Aviation Organization (ICAO) to categorize aircraft accidents and incidents. The taxonomy is used to collect and analyze data on aviation accidents and incidents, and to identify trends and patterns that can be used to improve aviation safety.

The ADREP taxonomy contains over 1000 terms,



ECCAIRS Aviation 1.3.0.12	V4 CD Damage aircraft
The aircraft was destroyed in the accident. (Destroyed)	1
The damage sustained makes it inadvisable to restore the aircraft to an airworthy condition. Note: This differs from the definition of a hull loss which reads: The aircraft is damaged beyond economical repair. A determination of "Hull loss" is thus not the result of a technical evaluation but may result from economic considerations.	
The aircraft sustained substantial damage in the accident. (Substantial)	2
The aircraft sustained damage or structural failure which: - adversely affected the structural strength, performance or flight characteristics of the aircraft and - would normally require major repair or replacement of the affected component, except for engine failure or damage, when the damage is limited to the engine, its cowlings or accessories; or for damage limited to propellers, wing tips, antennas, tyres, brakes, fairings, small dents or puncture holes in the aircraft skin. ICAO Annex 13. Major repair: a repair - (1) That, if improperly done, might appreciably affect weight, balance, structural strength, performance, powerplant operation, flight characteristics, or other qualities affecting airworthiness; or - (2) That is not done according to accepted practices or cannot be done by elementary operations.	
The aircraft sustained minor damage in the occurrence. (Minor)	3
Minor damage: The aircraft can be rendered airworthy by simple repairs or replacement and an extensive inspection is not necessary.	
The aircraft sustained no damage in the occurrence. (None)	98
The extent of the damage that the aircraft sustained in the occurrence is not known. (Unknown)	99

Figure 26: Attribute values about aircraft damage in ADERP. The damages are classified as destroyed, substantial, minor and none with explanation of each

.

which are organized into a hierarchy of categories and subcategories. The categories include

1. Accidents: Events that result in the death or serious injury of a person on board an aircraft, or the destruction of the aircraft.

2. Incidents: Events that do not result in death or serious injury, but that could have done so if not for the actions of the crew or other factors.

The subcategories within each category provide more detailed information about the accident or incident. For example, the subcategories for accidents include Aircraft category, Aviation operations, Damage aircraft, Geographical areas, Injury level, etc. Figure 26 shows different types of damages an aircraft can incur in an accident. The damages namely destroyed, substantial, minor and none are further explained in ADREP.

The ADREP taxonomy is used by ICAO member states to collect and analyze data on aviation accidents and incidents. The data is used to identify trends and patterns that can be used to improve aviation safety. For example, if a particular type of aircraft is involved in a high number of accidents, ICAO may recommend that the aircraft manufacturer take steps to improve the safety of the aircraft.

KG	Entities	Reln	Triples
AviationKG	15,137	151	193,372
Wikimovies	40,150	4	134,741
Wikidata5m	4,594,485	822	20,624,575
FB15k	14,951	1,345	592,213
FB15k-237	14,505	237	310,079
WordNet18	41,000	18	141,442

Table 2: Statistics of Benchmark Knowledge Graphs

7.2 Benchmark Knowledge Graphs

A knowledge graph (KG) is a structured collection of data that represents entities and their relationships. KGs are used to store and organize information about the real world, and they can be used for a variety of tasks, such as question answering, natural language processing, and machine learning. In this section, we will discuss popularly used KGs for experiments from different domains and provide statistics on their sizes.

7.2.1 WordNet

WordNet is a lexical database that organizes words into sets of synonyms called synsets, each representing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations, such as "is-a" (hypernymy) and "part-of" (meronymy). Hypernymy and hyponymy are two of the most important semantic relations in WordNet. Hypernymy is a "is-a" relationship, and it indicates that one concept is a more general concept than another. For example, the word "apartment" is a hyponym of the word "dwelling," because apartment is a type of dwelling. Hyponymy is the opposite of hypernymy, and it indicates that one concept is a more specific concept than another. WordNet can be viewed as a knowledge graph (KG), which is a structured collection of information about entities and their relationships. In WordNet, the entities are words, and the relationships are the semantic relations between them. For example, the semantic relation "is-a" can be viewed as a relationship between two entities, where one entity is a more general concept and the other entity is a more specific concept. WN18 [3] is a benchmark dataset for evaluating the performance of systems that work with knowledge graphs. WN18 contains 18 relations, 40K entities and 151K triples. The triples in WN18 are extracted from WordNet, and they represent the semantic relations between words in WordNet. Following are few examples from WN18: The head and tail entities are in the format entity(dot)part of speech(dot)sense. Let us understand the meaning of the triple <future.n.01, hypernym, time.n.05>. The head entity, future.n.01, is a noun with the sense of "the time yet to come." The tail entity, time.n.05, is a noun with the sense of "the continuum of experience in which events pass from the future through the present to the past." The relation between the two entities is hypernym, which means that future.n.01 is a type of time.n.05. In other words, the future is a part of time. The sense means of each entity can be understood using Stanford's WordNet Search [4].

7.2.2 Freebase

Freebase was a large-scale knowledge base that was acquired by Google in 2010. Freebase contained information about a wide variety of topics, including people, places, things, and events. Freebase is no longer available, but its data has been used to create other KG datasets, such as Google Knowledge Graph. FB15k and FB15k-237 [5]are knowledge graph datasets that are based on the Freebase knowledge base. They are commonly used as benchmarks for evaluating the performance of knowledge graph embedding models. FB15k contains 592,213 triples with 14,951 entities and 1,345 relationships. FB15k-237 is a subset of FB15k that contains 237 relationships. This was done to reduce the number of inverse relations in the dataset, as it was found that a large number of test triplets could be obtained by inverting triplets in the training set. In the KG triple </m/01qscs , /award/award_nominee/award_nominations./award/award_nomination/award , /m/02x8n1n >, /m/01qscs is "Benicio del Toro" and /m/02x8n1n is "Independent Spirit Award for Best Supporting Male". The relation indicates it is an award nomination. So the triple represents that Benicio del Toro was nominated for Independent Spirit Award for Best Supporting Male.

7.2.3 Wikidata

Wikidata is a free and open knowledge base that anyone can edit. It contains information about

[3]https://github.com/simonepri/datasets-knowledge-embedding

[4]http://wordnetweb.princeton.edu/perl/webwn

[5]https://github.com/simonepri/datasets-knowledge-embedding

people, places, things, and events, and is used by a wide variety of applications, including search engines, virtual assistants, and educational tools. Wikidata5m[6] is a subset of Wikidata that contains 5 million entities and their associated properties. It was created by the MilaGraph team at the University of Montreal, and is used for research in knowledge graph embedding and natural language processing. Wikidata5M has 822 relations and 20 million KG triples.

The knowledge graph is stored in the triplet list format, where each line corresponds to a triple of entity, relation, and value. For example, the triple <Q22686, P39, Q11696> corresponds to the triple Donald Trump position held President of the United States. The corpus is a collection of documents, indexed by entity ID. Each document provides a description of the entity. For example, the document for Donald Trump is Q22686 Donald John Trump (born June 14, 1946) is the 45th and current president of the United States .... The aliases file lists the aliases for entities and relations. For example, the line Q22686 donnie trump 45th president of the united states Donald John Trump ... lists the aliases for Donald Trump.

7.2.4 WikiMovies

The WikiMovies Knowledge Graph [7] is a knowledge graph that represents information about movies, actors, directors, and other entities related to the film industry. It is based on the Wikidata knowledge graph, which is a free and open knowledge base that anyone can edit. The WikiMovies Knowledge Graph was created by the WikiMovies project, which is a collaborative effort to create a comprehensive database of information about movies.

The WikiMovies Knowledge Graph contains facts about movies, actors, directors, and other entities. These facts are stored in the form of triplets, which consist of an entity, a relation, and a value. For example, the triplet (The Shawshank Redemption, directed by, Frank Darabont) represents the fact that the movie The Shawshank Redemption was directed by Frank Darabont. The WikiMoviesKG is used along with MetaQA for benchmarking QA for movies domain.

[6]https://deepgraphlearning.github.io/project/wikidata5m

[7]https://research.facebook.com/downloads/babi/

7.2.5 AviationKG

AviationKG [8] is a Knowledge Graph from aviation domain. AviationKG is created using NTSB reports which contain safety information. The Knowledge Graph contains information about weather conditions and events that can lead to accidents. It also has pilot and aircraft information which was recorded after the accident. In the KG triple <AccidentNumber_SEA08FA005 , IsCausedBy , DIVERTEDATTENTION> means that the aircraft accident associated with accident number SEA08FA005 is caused by diverted attention. AviationKG has 193,372 triples, 15,137 entities, and 151 relations created from 4000 NTSB reports.

7.3 Question Answering

Question answering (QA) is a task of retrieving relevant information from a given text or dataset to answer a user's question. CWQ, MetaQA, and other are used to train and evaluate QA models. CWQ and MetaQA datasets can use a knowledge graph to assist them to answer in closed book scenario. While MetaQA is a multihop QA dataset from movies domain, movies are still closer to general domain. We propose two closed-book QA datasets, AviationQA is a dataset from the aviation domain and specifically contains information about accidents.

Dataset	Train	Validation	Test
MetaQA 1-hop	96,106	9,992	9,947
MetaQA 2-hop	118,980	14,872	14,872
MetaQA 3-hop	114,196	14,274	14,274
AviationQA	367,304	10,000	10,000
CWQ	61,619	3,519	3,531

Table 3: Statistics of QA Datasets

7.3.1 Complex Web Questions

ComplexWebQuestions (Talmor and Berant, 2018) is a dataset designed for answering complex questions that necessitate reasoning over multiple web snippets. It comprises 34,689 examples, each consisting of a complex question, answers (including aliases), an average of 366.8 snippets per question, and a SPARQL query against Freebase. Here are a few illustrative questions from the dataset:

[8]https://github.com/ankush9812/Knowledge-Infusion-in-LM-for-QA

1. The actress that had the role of Martha Alston, plays what role in Finding Nemo?
2. Which school that Sir Ernest Rutherford attended has the latest founding date?
3. What movies does Leo Howard play in and that is 113.0 minutes long?
4. Where is the end of the river that originates in Shannon Pot?

The answer to the first question is "Dory". The actress who played Martha Alston is Ellen DeGeneres. She also provided the voice of Dory in the Disney/Pixar animated film, Finding Nemo.

7.3.2 MetaQA

The MetaQA dataset is a large-scale, multi-hop knowledge graph question answering (KGQA) dataset in the movie domain. It contains over 400,000 questions, which are divided into three levels of difficulty: 1-hop, 2-hop, and 3-hop questions. The 1-hop questions can be answered by looking up a single entity in the knowledge graph, while the 2-hop and 3-hop questions require multiple hops through the knowledge graph to answer.

The MetaQA dataset was created by (Zhang et al., 2018) by extending the MovieQA dataset. The MovieQA dataset contains over 100,000 questions, but it is limited to 1-hop questions. The MetaQA dataset was created by adding 300,000 2-hop and 100,000 3-hop questions to the MovieQA dataset. The questions in MetaQA can be answering using the knowledge graph, Wikimovies discussed in section 7.2.4.

7.3.3 AviationQA

AviationQA [9] (Agarwal et al., 2022b) is a 1 million factoid QA pairs for the aviation domain. The QA pairs were created by web scraping 12k reports from the National Transportation Safety Board (NTSB) website from 2009-2022. A set of 90 question templates was prepared using the common structure of documents in the format:

1. Where did the accident [ ] take place?
2. What is the model/series of the aircraft bearing accident number [ ]?
3. Was there fire on the aircraft of the accident number [ ]?
4. Was there fire on the aircraft of the accident number [ ]?

A question template was devised, and answers corresponding to these questions were extracted from each NTSB report. To identify the specific report associated with each question, we utilized brackets [ ] within the template, such as CHI07LA273 or LAX07LA148. NTSB reports consist of a semi-structured format, comprising unstructured data in paragraphs as well as structured data in tabular form. We employed a regular expression-based method to extract answers from each report based on the template. Subsequently, we thoroughly examined the generated question-answer pairs. Since the structure of some reports varied, separate scripts were developed to retrieve answers from those specific reports.

8 Evaluation Metrics

Performance is evaluated using several metrics for Question Answering (QA) and Information Retrieval systems. Exact Match (EM) checks if the predicted answer matches the ground truth exactly. Semantic Match focuses on the meaning alignment between predicted and ground truth answers. Precision measures accuracy, while Recall assesses the system's ability to retrieve relevant answers. Mean Rank evaluates the average rank of the correct answer, and Mean Reciprocal Rank (MRR) calculates the average reciprocal rank of the correct answer. Hits measure how often the correct answer appears within the top-N ranked answers. These metrics collectively assess QA system performance.

8.1 Exact Match and Semantic Match

Exact match and semantic match are two different ways of matching text. While exact match checks that the strings are same, semantic match checks if the meaning is the same. Both metrics are explained below.

8.1.1 Exact Match

Exact match is when the text in the query matches exactly the text in the document. Exact match is a simple and straightforward way to match text, but it can be limited in its ability to find relevant documents. For example, if the query is "What is the capital of France?", an exact match would only return documents that contain the exact phrase "What is the capital of France?". However, there

[9] https://huggingface.co/datasets/sakharamg/AviationQA

may be other documents that contain the same information, but in different words, such as “Paris is the capital of France”.
In the scenario with multiple gold answers, (Rajpurkar et al., 2016) defines Exact Match as the percentage of predictions that match any of the ground truth answers exactly.

8.1.2 Semantic Match

Semantic match is when the text in the query has the same meaning as the text in the document, even if the words are not exactly the same. Semantic match is a more sophisticated way to match text that can overcome some of the limitations of exact match. Semantic match uses natural language processing (NLP) techniques to understand the meaning of the text in the query and the document. This allows semantic match to find relevant documents that do not use exactly the same words as the query.

8.2 Precision, Recall and F1

Precision, recall, and F1 score are commonly used evaluation metrics in Question Answering (QA) systems to measure their performance. These metrics help assess the accuracy and completeness of the generated answers. Let’s delve into each metric, provide their formulas, and offer examples to illustrate their calculation.

8.2.1 Precision

Precision is a measure of how accurate a system is. It is calculated as the number of correct answers divided by the total number of answers returned by the system. For example, if a system returns 10 answers, and 8 of them are correct, then the precision is 0.8.

$$Precision = \frac{TP}{TP + FP}, \tag{5}$$

where TP is true positive and FP is false positive. A high precision indicates that the system is returning a lot of correct answers.
Precision@k is a variant of precision used to evaluate the performance of ranking systems. It is calculated by counting the number of relevant documents in the top k positions of a ranked list, divided by the total number of documents in the ranked list. A higher precision@k indicates better performance. The formula for precision@k is as follows:

$$Precision@k = \frac{topk}{total}, \tag{6}$$

where k is the number of positions considered, topk is the number of documents in the top k positions that are relevant to the query and total is the total number of documents in the ranked list.
For example, if there are 10 documents in a ranked list and 5 of them are relevant to the query, then the precision@5 would be 0.5.
Precision@k is a useful metric for evaluating the performance of ranking systems, as it considers the position of relevant documents in the ranked list. However, it is important to note that precision@k is sensitive to the number of relevant documents in the ranked list. A ranking system with high precision@k on a dataset with few relevant documents may not perform as well on a dataset with many relevant documents.

8.2.2 Recall

Recall is a measure of how complete a system is. It is calculated as the number of correct answers divided by the total number of correct answers. For example, if there are 10 correct answers, and a system returns 8, then the recall is 0.8.

$$Recall = \frac{TP}{TP + FN}, \tag{7}$$

where TP is true positive, and FN is false negative. A high recall indicates that the system is returning a lot of the correct answers.

8.2.3 F1

The F1 score is a measure of both precision and recall. It is calculated as the harmonic mean of precision and recall. The harmonic mean is a more sensitive measure of performance than the arithmetic mean because it gives more weight to low values. For example, if the precision is 0.8 and the recall is 0.6, then the F1 score is 0.72.

$$F1 = \frac{2 * (precision * recall)}{(precision + recall)}. \tag{8}$$

A high F1 score indicates that the system returns a good balance of correct answers and recall.

8.3 Mean Rank, MRR, and Hits@k

Mean Rank, Mean Reciprocal Rank (MRR) and Hits are three common metrics used to evaluate the performance of ranking algorithms.

8.3.1 Mean Rank

Mean rank is the average rank of all relevant documents in a ranked list. A lower mean rank indicates better performance. For example, a mean rank of 1 indicates that all relevant documents are at the top of the ranked list, while a mean rank of 10 indicates that all relevant documents are at the bottom of the ranked list. The formula for mean rank is:

$$Meanrank = \frac{1}{n}\sum rank, \tag{9}$$

where rank is the rank of each relevant document and n is the number of relevant documents. While MR is simple, it can be sensitive to the number of relevant documents in a ranked list. Let us say that a few relevant documents are ranked very high due to errors in the ranking system. It will pull down the average. Also, it gives importance to lower ranks. Many a time, for a system 20th rank, would be equally bad as 100th rank. At the same time, ranks 1,3,5,7, and 10, even though they are close, would make a huge difference while evaluating a ranking system.

8.3.2 Mean Reciprocal Rank (MRR)

Mean reciprocal rank is the average of the reciprocal ranks of all relevant documents in a ranked list. A higher mean reciprocal rank indicates better performance. For example, a mean reciprocal rank of 1 indicates that all relevant documents are at the top of the ranked list, while a mean reciprocal rank of 0.5 indicates that the relevant documents are ranked in the middle of the ranked list. The formula for MRR is:

$$MRR = \frac{1}{n}\sum \frac{1}{rank}, \tag{10}$$

where rank is the rank of each relevant document and n is the number of relevant documents. MRR gives higher importance to the top ranks and lower importance to the bottom ranks. In the case of MRR, rank 1 and rank 2 has a difference of 0.5 while rank 10 and 100 have a difference of 0.09. MRR is very useful in QA systems as the correct answers to a question are only a few and when ranked correctly we should ignore ranks away from top ranks.

8.3.3 Hits@k

Hits@k is the percentage of queries for which at least one relevant document is ranked in the top k positions. A higher hits@k indicates better performance. For example, a hits@k of 1 indicates that all queries returned at least one relevant document in the top k positions, while a hits@k of 0.5 indicates that half of the queries returned at least one relevant document in the top k positions. The formula for hits@k is:

$$Hits@k = \frac{1}{n}\sum[1 \text{ if } rank \leq k \text{ else } 0] \tag{11}$$

where rank is the rank of each relevant document and n is the number of queries. For example, if there are 5 queries and 2 relevant documents are ranked in the top 3 positions, then the hits@3 would be 2/5 = 0.4. Hits@k is useful in QA systems when we want the answer to a question at top-k ranks. When a QA model predicts multiple answers, a gold answer might be at 2nd rank, but 1st rank answer could be another gold answer. In such situations, we don't want to penalize our model. Usually when evaluating a system, hits are calculated with different values of k and can be helpful in identifying how many top results should be shown to ensure that the user gets the answer to its question most of the time with less lookup through the k results.

9 Summary

This paper provided an overview of knowledge graphs, deep learning for NLP and information retrieval techniques. It also discussed question-answering methods followed by ongoing research on knowledge-infused QA, including the CFILT, IIT Bombay research. Finally, we explained different evaluation metrics for QA and retrieval systems.

10 Conclusion and Future Work

In this survey paper, we have explored the concept of knowledge graphs and their infusion techniques. Knowledge graphs have emerged as a powerful framework for representing structured knowledge and capturing relationships between entities. They offer a holistic view of information, enabling efficient reasoning, search, and data integration. We discussed various approaches for constructing knowledge graphs, ranging from manual curation to automated extraction techniques, along with popular construction frameworks and tools. Looking ahead, several avenues for future work in the field of knowledge graphs and infusion present exciting opportunities. Firstly, research should focus on developing scalable techniques for handling large-scale knowledge graphs efficiently. As the size and complexity of knowledge graphs continue to grow, efficient storage, querying, and reasoning mechanisms are essential. Secondly, improving the quality and reliability of integrated data remains a critical challenge. Addressing multilingual KGs, data heterogeneity, ensuring data accuracy, and applying quality control measures during the infusion process is vital for maintaining the integrity of knowledge graphs. Furthermore, exploring techniques to incorporate temporal and dynamic aspects into knowledge graphs would enhance their ability to capture evolving relationships and enable more accurate predictions and recommendations.

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.

Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. 2021. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3554–3565, Online. Association for Computational Linguistics.

Hannah Bast and Elmar Haussmann. 2015. More accurate question answering on freebase. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pages 1431–1440.

Pushpak Bhattacharyya. 2022. Cs772: Deep learning for natural language processing.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

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.

Niel Chah. 2017. Freebase-triples: A methodology for processing the freebase data dumps.

Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.

Jacob Cohen. 1968. Weighted kappa: nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213.

Wikimedia Commons. 2022. File:knowledgegraphembedding.png — wikimedia commons, the free media repository. [Online; accessed 13-June-2023].

Yuanfei Dai, Shiping Wang, Neal N. Xiong, and Wenzhong Guo. 2020. A survey on knowledge graph embedding: Approaches, applications and benchmarks. *Electronics*, 9(5).

Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. 2018. Hyte: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2001–2011.

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.

J. Guo, Z. Wang, Y. He, J. Su, and Y. Yu. 2020. A survey on rule-based reasoning for knowledge graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(4):1–32.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020a. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020b. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Yongqun He, Hong Yu, Edison Ong, Yang Wang, Yingtong Liu, Anthony Huffman, Hsin-hui Huang, John Beverley, Junguk Hur, Xiaolin Yang, et al. 2020. Cido, a community-based ontology for coronavirus disease knowledge and data integration, sharing, and analysis. *Scientific data*, 7(1):181.

Marti A Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *COLING 1992 Volume 2: The 14th International Conference on Computational Linguistics*.

Henry Hsu and Peter A Lachenbruch. 2014. Paired t test. *Wiley StatsRef: statistics reference online*.

Jin Huang, TingHua Zhang, Jia Zhu, Weihao Yu, Yong Tang, and Yang He. 2021. A deep embedding model for knowledge graph completion based on attention mechanism. *Neural Computing and Applications*, 33(15):9751–9760.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696, Beijing, China. Association for Computational Linguistics.

Kaveri Kale, Pushpak Bhattacharyya, Milind Gune, Aditya Shetty, and Rustom Lawyer. 2023. KGVL-BART: Knowledge graph augmented visual language BART for radiology report generation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3401–3411, Dubrovnik, Croatia. Association for Computational Linguistics.

Kaveri Kale, Pushpak Bhattacharyya, Aditya Shetty, Milind Gune, Kush Shrivastava, Rustom Lawyer, and Spriha Biswas. 2022. Knowledge graph construction and its application in automatic radiology report generation from radiologist's dictation.

Omar Khattab and Matei Zaharia. 2020. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 39–48, New York, NY, USA. Association for Computing Machinery.

Douglas B. Lenat. 1995. Cyc: A large-scale investment in knowledge infrastructure. *Commun. ACM*, 38(11):33–38.

Belinda Z. Li, Sewon Min, Srinivasan Iyer, Yashar Mehdad, and Wen-tau Yih. 2020. Efficient one-pass end-to-end entity linking for questions. In *EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*.

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 *AAAI*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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.

George A. Miller. 1992. WordNet: A lexical database for English. In *Speech and Natural Language: Proceedings of a Workshop Held at Harriman, New York, February 23-26, 1992*.

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.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, Madeleine Thompson, Tabarak Khan, Toki Sherbakov, Joanne Jang, Peter Welinder, and Lilian Weng. 2022. Text and code embeddings by contrastive pre-training.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 809–816, Madison, WI, USA. Omnipress.

OpenAI. 2023. Chatgpt. Accessed on May 31, 2023.

Jay Pujara, Eriq Augustine, and Lise Getoor. 2017. Sparsity and noise: Where knowledge graph embeddings fall short. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1751–1756, Copenhagen, Denmark. Association for Computational Linguistics.

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. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

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.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, Seattle, United States. 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.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Noam M. Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *ArXiv*, abs/1804.04235.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4444–4451. AAAI Press.

Haitian Sun, Tania Bedrax-Weiss, and William Cohen. 2019. PullNet: Open domain question answering with iterative retrieval on knowledge bases and text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2380–2390, Hong Kong, China. Association for Computational Linguistics.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. 2021. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*.

A. Talmor and J. Berant. 2018. The web as a knowledge-base for answering complex questions. In *North American Association for Computational Linguistics (NAACL)*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Gerhard Weikum, Xin Luna Dong, Simon Razniewski, Fabian Suchanek, et al. 2021. Machine knowledge: Creation and curation of comprehensive knowledge bases. *Foundations and Trends® in Databases*, 10(2-4):108–490.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. 2020. Scalable zero-shot entity linking with dense entity retrieval.

Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

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*.

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