Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 6260–6270
Marseille, 20-25 June 2022

Knowledge Graph – Deep Learning: A Case Study in Question Answering in Aviation Safety Domain

Ankush Agarwal[1], **Raj Gite**[1], **Shreya Laddha**[1], **Pushpak Bhattacharyya**[1], **Satyanarayan Kar**[2], **Asif Ekbal**[3], **Prabhjit Thind**[2], **Rajesh Zele**[1], **Ravi Shankar**[2]
[1]IIT Bombay, [2]Honeywell, [3]IIT Patna
{ankushagrawal, rajgite, pb}@cse.iitb.ac.in, {shreyaladdha, rajeshzele}@ee.iitb.ac.in, asif@iitp.ac.in,
{satya.kar, prabhjit.thind, ravishankar.r}@honeywell.com

Abstract

In the commercial aviation domain, there are a large number of documents, like accident reports of NTSB and ASRS, and regulatory directives ADs. There is a need for a system to efficiently access these diverse repositories to serve the demands of the aviation industry, such as maintenance, compliance, and safety. In this paper, we propose a Knowledge Graph (KG) guided Deep Learning (DL) based Question Answering (QA) system to cater to these requirements. We construct a KG from aircraft accident reports and contribute this resource to the community of researchers. The efficacy of this resource is tested and proved by the proposed QA system. Questions in Natural Language are converted into SPARQL (the interface language of the RDF graph database) queries and are answered from the KG. On the DL side, we examine two different QA models, BERT-QA and GPT3-QA, covering the two paradigms of answer formulation in QA. We evaluate our system on a set of handcrafted queries curated from the accident reports. Our hybrid KG + DL QA system, KGQA + BERT-QA, achieves 7% and 40.3% increase in accuracy over KGQA and BERT-QA systems respectively. Similarly, the other combined system, KGQA + GPT3-QA, achieves 29.3% and 9.3% increase in accuracy over KGQA and GPT3-QA systems respectively. Thus, we infer that the combination of KG and DL is better than either KG or DL individually for QA, at least in our chosen domain.

Keywords: Question Answering, Knowledge Discovery/Representation, Information Extraction, Information Retrieval

1. Introduction

An extensive set of documents is used in the aerospace sector, *viz.*, system descriptions, manuals, and procedures. In addition to the industrial technical manuals, which have limited accessibility, there is a wide range of publicly available datasets related to aircraft safety. The majority of these datasets are subject to specific regulations or procedures, and safety professionals utilize them to investigate accidents and trends in aviation safety. Naturally, a Question Answering system is desired due to the large number and extensive length of such documents. For example, when investigating an accident, the ability to query the documents is required to locate similar events or find rare occurrences.

In this paper, we present an Aviation Knowledge Graph, constructed from the accident reports, to help safety experts study the reported accidents. It contains detailed information about the accidents' cause, effect, aircraft, pilot, and meteorological conditions. KGs provides a way to integrate and relate a large number of scattered, isolated pieces of information, thus converting it into knowledge that supports making complex inferences. However, domain expertise is required to construct a KG. Besides, KGs are not as expressive as natural language sentences to explain complex events. This motivates the need for Deep Learning models such as BERT (Devlin et al., 2018) from Google, and GPT-3 (Brown et al., 2020) from Open AI, which can fetch full-length answers from unstructured text in the documents. Our goal in this study is to combine the high coverage of textual corpora with the entity relationships in KG to improve the retrieval coverage and accuracy of the overall system.

In this paper, we show that such an architecture relying on both Knowledge Graph and Deep Learning can benefit Question Answering. Our **major contributions** in this paper are:

1. We introduce an **Aviation Knowledge Graph**, constructed from several accident reports using the domain knowledge and information extraction techniques.

2. We propose a **Question Answering system** that answers the questions asked in natural language from two modules – (a) KG-based QA (KGQA) module, which uses a pipeline to convert Natural Language Queries to SPARQL queries and fetches responses from the Aviation Knowledge Graph, and (b) DL-based QA (DLQA) module that extracts answers from the plain text in the documents. The DLQA module has been tested using two different QA models, BERT-QA and GPT3-QA.

3. We show that a **combined Question Answering system**, such as ours, outperforms the individual Knowledge Graph and Deep Learning meth-

ods on our curated test set. The combined system KGQA + BERT-QA attains **7%** and **40.3%** increase in accuracy over KGQA and BERT-QA modules respectively. Similarly, the other hybrid system KGQA + GPT3-QA attains **29.3%** and **9.3%** increase over KGQA and GPT3-QA modules respectively. Hence, we provide strong evidence that the two modules, KGQA and DLQA, complement each other, and neither of them is dispensable for the task of QA.

This paper is organized as follows. In Section 2, we present a brief survey of the literature. Section 3 describes the public data repositories in the aviation domain relevant for building a QA system. Section 4 shows the details for constructing the Aviation Knowledge Graph and a brief explanation of its properties. Section 5 explains the overall system architecture used for Question Answering. Section 6 discusses the models, test datasets, and metrics used for evaluation. A summary of the main findings is presented in Section 7. We conclude with a direction to future work in Section 8.

2. Related Work

Knowledge Graph construction in the aviation field has been an important research topic. Zhao et al. (2018) shows the development of a KG construction system in the aviation risk field, where the American Aviation Safety Reporting System (ASRS) reports are used as an example to verify the rationality and validity of the KG construction method. Cheng et al. (2019) built an ontology for civil aviation security by extracting knowledge from the data sources in the form of entities and relations. Wang et al. (2020) also discusses method of extracting entities and relations from structured and unstructured text in aviation domain. However, no prior work has been done on the widely available NTSB accident reports. We exploit this gap and construct a KG from the aircraft accident reports for aviation safety.

Querying Knowledge Graphs in natural language has been a long-standing research challenge. Early work focused on rule-based, and pattern-based systems (Affolter et al., 2019) for the Text-to-SQL task, which later moved towards using seq2seq architecture (Zhong et al., 2017) and pre-trained models (He et al., 2019) with the advent of Neural Networks (NN). We are interested in translating Natural Language Queries to SPARQL – the standardized query language for RDF graph databases. Several QA systems have been developed that use rule-based approaches (Diefenbach et al., 2017) to answer questions over DBpedia and Wikidata. Some of the prior research (Singh et al., 2018) divides the whole QA pipeline into smaller sub-tasks while others (Diefenbach et al., 2020) combine several components to make a pipeline. We use an approach similar to Liang et al. (2021) in which the translation task is divided into KG-dependent and KG-independent sub-tasks.

Question Answering using Deep Learning has been a widely explored area in general. However, not much progress has been made in the aviation domain due to the frequently occurring in-domain technical jargon. A few existing works use large pre-trained transformer-based language models for Question Answering. Kierszbaum and Lapasset (2020) use Distilled BERT for Question Answering on ASRS reports for a small set of documents and limited test data. Arnold et al. (2020) employs a BM25-based retriever, followed by BERT fine-tuned for QA on a general domain data set. This model is used as a baseline for benchmarking our QA system.

Our work is the first attempt to combine Knowledge Graph and Deep Learning in the aviation domain for Question Answering to the best of our knowledge.

3. Data Repositories

Multiple organizations investigate aircraft accidents and warehouse all the details of casualties in a database. We identify four such public repositories, and demonstrate the use of NTSB accident reports and ADREP taxonomy in this case study. The recognized databases are the following:

- **Accident reports** capture all the aspects of an accident, namely aircraft specifications, pilot details, environmental state, and a comprehensive description of the suspected cause of the accident. We use two publicly available accident repositories – **National Transportation Safety Board (NTSB) reports**[1] (a sample report is shown in Figure 3 in Appendix) and **Aviation Safety Reporting System (ASRS) reports**[2]. The NTSB stores investigation reports of civil aviation accidents in the US, whereas ASRS gathers information from pilots and crew members about close call events during flight journeys. These documents contain information in structured as well as unstructured format.
- **Airworthiness Directives (ADs)**[3] are notifications to owners of certified aircrafts about the unsafe conditions that exist in particular aircraft models and the corresponding corrective measures, which are absent in the accident reports. In our work, ADs issued by the Federal Aviation Administration (FAA) of the United States are considered.

[1]`https://www.ntsb.gov/Pages/AviationQuery.aspx`

[2]`https://asrs.arc.nasa.gov/search/reportsets.html`

[3]`https://www.faa.gov/regulations_policies/airworthiness_directives/`

- **Accident/Incident Data Reporting (ADREP) Taxonomy**[4] (a part of the taxonomy is shown in Figure 4 in Appendix) contains a complex multi-level hierarchy of factual descriptors (time, place, aircraft models, engine, component manufacturers, etc.) and analytical descriptors of the occurrence of accident, such as event types and explanatory factors. It is especially helpful in the construction of ontology.

4. Aviation Knowledge Graph

We construct an Aviation Knowledge Graph using NTSB reports and ADREP taxonomy in Protégé (Musen, 2015). A total of 4000 NTSB reports from 1962 to 2015, having an average of 3000 words per report, are used in the construction. The NTSB reports consist of paragraphs (unstructured) and tables (structured) containing the information about aircraft accidents. This section discusses the pre-processing of NTSB reports, ontology creation, Knowledge Graph construction, KG evaluation, and challenges in the creation of Aviation KG. Figure 1 shows the pipeline for building a KG from NTSB reports.

4.1. Pre-processing

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

4.2. Ontology Creation

With the help of domain experts and ADREP taxonomy, we construct an ontology from the accident reports. For example, domain knowledge from ADREP 'Events' taxonomy (a snippet is shown in Figure 4 in Appendix) is used for creating the *Event* class. 'Events' taxonomy is relevant because every accident report has an event sequence that gives a gist of the cause. For ontology creation, we extract the ADREP taxonomy in a tree-like data structure such that a unique path exists from the root to each leaf node. Subsequently, we obtain classes by mapping NTSB occurrences to the adequate root-to-leaf paths. We use the following two techniques for mapping:

- **Mapping based on the distance between BERT embeddings**: We obtain embeddings for each path in the ADREP taxonomy and NTSB events using the BERT model. The distance is calculated between ADREP and NTSB embeddings, and the ADREP node with least distance is mapped to the NTSB event. After that, the corresponding root-to-leaf path in the ADREP tree hierarchy is associated with the NTSB event.

- **Mapping based on Keyword Matching**: We tokenize the NTSB report events and map each token to the nodes in the ADREP taxonomy. The node with the highest similarity score is associated with the NTSB event.

Following is an example of an ADREP event mapped to NTSB occurrence (more detailed ADREP hierarchy is shown in Figure 4).

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

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

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

- **Named Entity Recognition (NER)** (Nadeau and Sekine, 2007) is used to identify names, organizations, etc., from the NTSB text. These named entities are inserted as entity classes in our Ontology. For example, Long Beach, Las Vegas, and Chicago are the instances of *Location* class present in NTSB reports identified by the NER method.

- **Term Frequency – Inverse Document Frequency (TF-IDF)** (Juan, 2003) technique is used to determine the important terms across NTSB documents, which are later organized as entity classes in Ontology. For example, Aircraft ID, Aircraft Damage, and Pilot Certificate are some of the important terms identified by the TF-IDF technique.

- **C-value** (Frantzi et al., 2000) overcomes the drawback of TF-IDF in obtaining multi-word terms. Multi-words are necessary to identify the entities such as Landing Gear, Vertical Stabilizers, etc., in aircraft reports.

- **Sentence Clustering** (Lu and Fu, 1978) is used for grouping similar sentences and discovering the entities among them.

- **Syntactic Analysis** technique is used where the corpus is annotated with part-of-speech tags to

[4] `https://www.icao.int/safety/airnavigation/aig/pages/adrep-taxonomies.aspx`

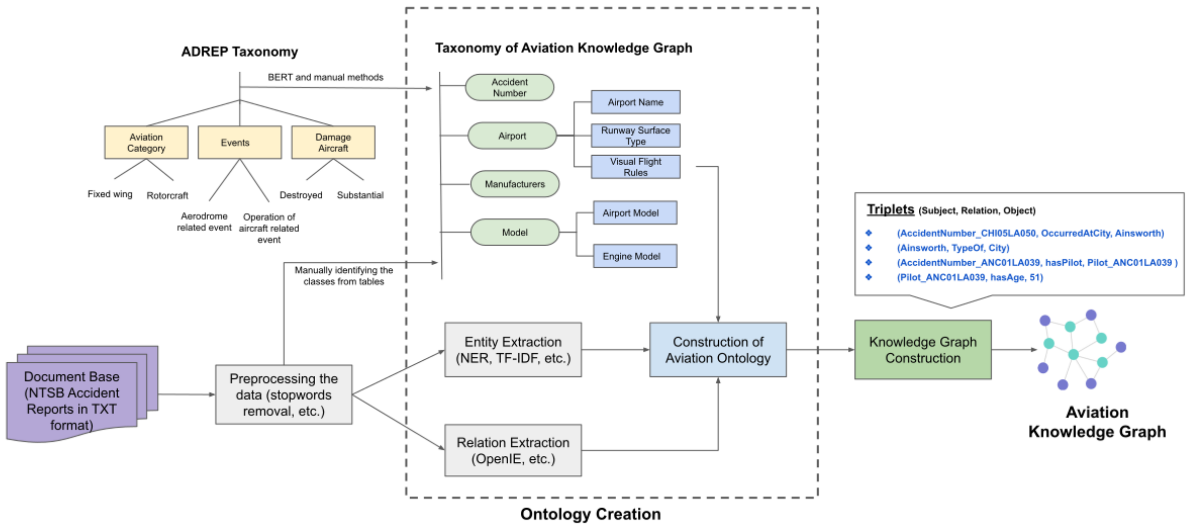

ADREP Taxonomy
BERT and manual methods
Aviation Category
Events
Damage Aircraft
Fixed wing
Rotorcraft
Aerodrome related event
Operation of aircraft related event
Destroyed
Substantial
Taxonomy of Aviation Knowledge Graph
Accident Number
Airport
Airport Name
Runway Surface Type
Visual Flight Rules
Manufacturers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

Figure 1: Aviation Knowledge Graph Construction Process from NTSB reports

find hypernyms, hyponyms, and meronyms. We extract all the sentences containing three or more nouns in the analysis. We then manually try to identify lexically related entities. The intuition behind selecting nouns is that most of the time, the entities (classes and instances) are tagged as nouns in the ontology. The examples of such findings are – (a) 'Scratches are found on engine's nacelle.' From this, we can observe that *nacelle* is part of *engine*. (b) 'During the engine inspection, it was observed that the wing mounted engine was working, but the fin mounted engine was not working.' Here, we observe that *wing mounted engine* and *fin mounted engine* are the types of *engine*.

- **DL based techniques** are also used for entity extraction such as NER_DL [5] (trained on CoNLL dataset (Sang and De Meulder, 2003) and uses GloVe word embeddings (Pennington et al., 2014)), Onto_100 and NER_DL_BERT (trained on CoNLL dataset and uses BERT embeddings).

Dependency Analysis (Fundel et al., 2007) and **OpenIE** [6] techniques are used for relation extraction. The extracted relations are observed and inserted as properties in our Aviation Ontology. Furthermore, we manually add some properties by observing the entity classes in Ontology from NTSB reports.

4.3. Knowledge Graph Construction

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

An example for formation of triplets is as follows: *'Directional control - Not attained'* – It is an 'Aircraft Issue' present in the 'Findings' section of NTSB report in tabular format (see Figure 3 in Appendix). In our Aviation Ontology, 'Directional control' is an instance present in *Aircraft_cause* class, and 'Not attained' is an instance in *Aircraft_cause_reason* class. The triples (subject, relation, object) formed from this snippet are – **{Accident_Number, isCausedByAircraftIssue, Directional control}**, **{Directional control, isCausedDueTo-AircraftIssue, Not attained}** *Accident_Number* is the entity class in our Aviation Ontology containing the unique accident numbers of all NTSB reports as instances. *Accident_Number* class is an essential point source for our question answering system, and thus, it is linked with *Aircraft_cause* class using relation *isCausedByAircraftIssue*. In the second triplet, a relation, *isCausedDueTo-AircraftIssue*, is formed for linking the *Aircraft_cause* class with *Aircraft_cause_reason* class.

Table 1 describes the properties for Aviation Knowledge Graph constructed from the NTSB reports in Protégé. The total size of file containing the Knowledge Graph is around 12 MB.

4.4. Knowledge Graph Evaluation

We manually evaluate the terms obtained through entity and relation extraction techniques. A domain expert in Honeywell Corporation provided a small set of cases to validate the reach of the constructed Knowledge Graph. A total of 120 SPARQL queries with gold answers of different categories were tested, where our KG answered 83 questions, thereby achieving an accuracy of 69.2%.

[5] `https://deeppavlov.ai/`

[6] `https://nlp.stanford.edu/software/openie.html`

4.5. Challenges in Construction of Aviation Knowledge Graph

The main challenge we faced with Aviation Knowledge Graph is scalability. As explained previously, we use many different techniques for extracting entities and relations. Later, we require manual processing for creating Ontology from these extracted entities. Similarly, extraction of triplets from NTSB reports requires observing patterns in reports which may change with new reports. Another challenge we observed is that the domain specific terms or keywords present in NTSB reports are not identified effectively by the extraction models.

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

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

5. The Question Answering System

Knowledge Graph guided Deep Learning based QA system is composed of two modules – KG-based QA (KGQA) and DL-based QA (DLQA). The KGQA utilizes Aviation KG, while DLQA uses the text in the NTSB accident reports. These two modules work in parallel to answer questions as shown in Figure 2.

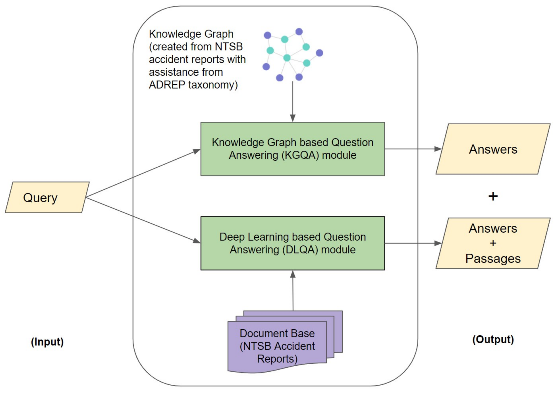

Knowledge Graph (created from NTSB accident reports with assistance from ADREP taxonomy)
Knowledge Graph based Question Answering (KGQA) module
Deep Learning based Question Answering (DLQA) module
Query
Answers
+
Answers + Passages
Document Base (NTSB Accident Reports)
(Input)
(Output)

Figure 2: Knowledge Graph guided Deep Learning based Question Answering system

The input to the QA system is a question in natural language, which is fed to both the modules, and the output of the QA system is a combination of responses from the two modules such that the output of DLQA module follows that of the KGQA module. The output of the KGQA module is a list of answers – entities, relations, or properties of the Aviation Knowledge Graph. DLQA module returns answers and passages whose specifications depend on the underlying model. If BERT-QA is used, we obtain a list of answer-passage pairs as response where each answer is a text span extracted from the corresponding passage. GPT3-QA returns a list of passages and an answer derived from the passage list. Figure 5 (in Appendix) shows a sample output from KGQA and DLQA modules. In our work, we give equal importance to both answers and passages because passages help the user when answers fetched by the QA system are incorrect.

Our proposal of combining KG and DL is based on the following rationale:

- KGQA has the advantage of domain knowledge present in the KG that captures essential details and helps in answering complex questions correctly and completely.
- DLQA lacks domain knowledge, but it has the advantage of coverage. With DLQA, all the text in the domain is considered for QA. This allows DLQA to answer questions that KGQA is unable to answer since KG does not capture all the information from the text.

Section 7 provides empirical evidence that such a system formed with the synergy of Knowledge Graph and Deep Learning surpasses the performance of each individual module. In the following subsections, we discuss the design of these modules.

5.1. KGQA module

For querying Knowledge Graph stored in Protégé, we need to convert the Natural Language Query into a SPARQL query. We adapt an approach similar to Liang et al. (2021) for our NL2SPARQL pipeline (see Figure 7 in Appendix) which consists of the following:

- **Question Type Classification**: We classify the questions into List/Boolean/Count categories to identify the keyword from DISTINCT/ASK/COUNT to use in the SELECT clause and then construct the WHERE clause in the SPARQL query.
- **Entity-Relation Extraction**: We implement techniques such as PoS tagging, Tokenization & Compounding, N-gram tiling, and dependency parsing for extracting entities and relations from the question. Then, we compute the similarity scores between the BERT embeddings to map the retrieved entities and relations to the corresponding entries in the underlying Knowledge Graph. This sub-task is the only KG-dependent sub-task in the pipeline.
- **Triple Generation and Ranking**: RDF triples are constructed using all the permutations of the mapped entities and relations. However, only

the valid ones are retained by performing a quick check with the Knowledge Graph by executing an ASK query. We then conduct a ranking step based on the syntactic similarity between the triples and the input question.

- **Query Construction**: SPARQL queries are generated using the keyword specifying question type (DISTINCT for List, ASK for Boolean, COUNT for Count) and the triples obtained from the previous step. Proper PREFIX is also defined, specifying the base Knowledge Graph used.

The constructed SPARQL queries are executed over the Aviation Knowledge Graph to retrieve answers from KG which are either subjects or objects in the triples. The KGQA does not produce any response when the pipeline cannot form a valid triple in the Triple Generation & Ranking sub-task.

5.2. DLQA module

We examine two models, namely BERT-QA and GPT3-QA (Brown et al., 2020), which form two independent DLQA modules. The reason for reviewing BERT-QA and GPT3-QA is to cover the two paradigms of answer formulation in QA, namely extractive and abstractive. Both the models require a collection of passages from the documents. BERT-QA extracts a text span from the relevant passage as an answer (extractive QA), while GPT3-QA generates text from the relevant passages (abstractive QA). Both these models are based on the retriever-reader pipeline as follows:

- **Retriever**: The retriever retrieves top-k relevant passages given a question. It projects the question and all the passages in the collection to a semantic vector space, such that passages relevant to a question are positioned closer to the question as compared to less relevant passages. This projection of query and passages into semantic vector space is accomplished by Sentence-BERT (Reimers and Gurevych, 2019a) in case of BERT-QA and by GPT3 'ada' model in case of GPT3-QA [7]. Passages relevant to the question obtain a higher relevance score, and top-k passages are passed to the reader.

- **Reader**: The reader extracts/generates answers from the relevant passages given a question. It is a Machine Reading Comprehension (MRC) task where the underlying model understands the context passage and tries to answer the given question. In BERT-QA, we employ BERT (Devlin et al., 2018) fine-tuned on MRC task as the reader, which extracts text spans from the passages as the answer. In GPT3-QA, we utilize GPT3 'curie' model as the reader, which generates text from the passages as an answer.

Figure 8 (in Appendix) shows the retriever-reader pipeline along with an example.

6. Experimental Setup

Data Pre-processing: The NTSB reports, available as PDFs, are converted to TXT format for constructing the Knowledge Graph (as mentioned in Section 4). Passages from these reports are extracted and stored as JSON objects with three properties – passage heading, passage text, and report ID. The resulting JSON files are used by the BERT-QA model. The reports are also converted to JSONL format, where each row contains the extracted passages. These JSONL files are consumed by the GPT3-QA model.

Models: The details of the models used in our Question Answering system are as follows:

- The `bert-base-nli-mean-tokens` [8] model from the Sentence Transformers (Reimers and Gurevych, 2019b) library is used for similarity calculation in the **KGQA** module.

- In the **BERT-QA** model, we use a fine-tuned Sentence BERT model, namely `multi-qa-MiniLM-L6-cos-v1`[9] from the Sentence Transformers library, as the retriever. This Sentence BERT model is fine-tuned for the Semantic Search task using 215M question-answer pairs. Similarly, a BERT model, `deepset/bert-base-cased-squad2`[10] from the Transformers library (Wolf et al., 2020), fine-tuned on MRC task using SQuAD 2.0 dataset (Rajpurkar et al., 2018) is used as the reader.

- A **baseline** QA system is built on a similar approach as BERT-QA, called **BM25-BERT**. In this model, BM25 (Robertson and Zaragoza, 2009) relevance score is used for retrieving most relevant passages using sparse lexical features like TF-IDF. The BERT model in BM25-BERT has the same configurations as the reader in the BERT-QA model.

- From various flavors of **GPT-3**, we chose a combination of *ada* and *curie* as retriever and reader respectively, in the GPT3-QA model, based on offline evaluations.

Our QA system, which comprises of KGQA and DLQA modules, is configured to output ten responses such that the outputs from the DLQA follow those from KGQA. When both the modules have sufficient outputs to display, each module returns its top five

[7] `https://beta.openai.com/docs/guides/answers`

[8] `https://huggingface.co/sentence-transformers/bert-base-nli-mean-tokens`

[9] `https://huggingface.co/sentence-transformers/multi-qa-MiniLM-L6-cos-v1`

[10] `https://huggingface.co/deepset/bert-base-cased-squad2`

results, else DLQA adjusts its return count to make up for the shortfall in the KGQA responses.

Test Dataset: For evaluating the performance of our QA system, a test set is curated from 50 NTSB reports. We restrict the number to 50 by considering the difficulty in incorporating all 50 reports while creating a single test instance. All these reports belong to the year 2002. A total of 150 test instances were created (a single instance of test set is shown in Figure 6 in Appendix), where each instance consists of a query, a list of actual answers, and a list of passages where the answers can potentially be found. The entire test set is manually curated by looking up every query in those 50 reports and recording the desired answers and the paragraphs. We tag each paragraph with the unique accident number in its report, which aids in evaluating the retriever sub-module of DLQA. We do not evaluate our QA system with standard QA datasets because the KG used in KGQA module is domain-specific, and thus, does not contribute when our QA system operates on other domains.

Evaluation Metrics: Our overall system outputs both answers and passages. Sometimes, the predicted answer may not be correct, so reading the passage allows the user to reasonably satisfy the query intent. Thus, both the accuracy of answers and passages become important, and we evaluate both separately. Finally, we report the performance using four metrics:

- **Exact Match** (binary value): An exact match occurs if the first answer predicted by the system is present in the list of actual answers exactly.
- **Exact Recall**: We determine exact recall as the fraction of actual answers available exactly in the top 10 answers predicted by the system. However, for more than 10 actual answers, we consider only a subset such that they contain the maximum number of predicted answers. This is done to ensure a recall ≤ 1 since we predict 10 answers only.
- **Semantic Accuracy** (binary value): We use this as a flexible alternative to Exact Match, which relies solely on lexical overlap. We define our output as semantically accurate if any value in the list of top 10 predicted targets is semantically similar to any value in the list of actual targets. This metric does not consider the rank of the predicted targets, unlike Exact Match.
- **Semantic Recall**: We compute this metric as the fraction of actual targets which are semantically similar to the top 10 predictions of the system. Similar to Exact Recall, we consider only a subset of 10 targets such that they contain the maximum number of predicted targets.

Exact Match and Exact Recall are calculated only for the answers predicted by the system, whereas Semantic Accuracy and Semantic Recall are computed for both the answers and the passages. Exact Match and Semantic Accuracy are a measure of correctness of our Question Answering system, whereas Exact Recall and Semantic Recall estimate the completeness of the outputs of our system.

7. Results and Analysis

This section analyzes the performance of 6 models – BM25-BERT (baseline), KGQA, BERT-QA, GPT3-QA, KGQA + BERT-QA, and KGQA + GPT3-QA. The last two models are the combined models that use both KG and DL, whereas the first four are individual models. We evaluate both the answers and passages predicted by these models. Table 2 presents the evaluation results of answers on Exact Match and Exact Recall metrics as well as the evaluation results of answers and passages on Semantic Accuracy and Semantic Recall metrics. As the KGQA system only outputs answers, we consider the answers from KGQA as passages when evaluating passages in KGQA + BERT-QA and KGQA + GPT3-QA models.

It is trivial to see that the Exact Match metric scores are relatively lower than the Semantic Accuracy metric scores because of the dependence of Exact Match on only the lexical aspect of answers. Thus, we are more interested in Semantic Accuracy and Semantic Recall values. Each of the KGQA, BERT-QA, and GPT3-QA models beat the baseline model on every metric. KGQA system performs better than the BERT-QA model, which validates the importance of domain knowledge. GPT3-QA model is the best performing among the individual models owing to its large size and extensive pre-training.

Our combined models perform better than their counterparts in most cases. Since the top answers from each system are merged to generate top-10 answers for the combined KG + DL system, accuracy and recall scores increase. However, due to the fact that our system outputs KGQA answers at the top of DLQA, the Exact Match score of KGQA + GPT3-QA is lower than that of GPT3-QA. In other scenarios, the KGQA + GPT3-QA system achieves an increase of 29.3% and 9.3% increase in answer accuracy over KGQA and GPT3-QA respectively. The KGQA + BERT-QA system also attains an increase in answer accuracy of 7% over KGQA and 40.3% over BERT-QA. A similar trend is seen in passage retrieval accuracy for both the hybrid QA models. Overall, **KGQA + GPT3-QA is the best model for the QA task**.

Limitations of KGQA module: The low accuracy of the KGQA model can be attributed to the following factors – (a) Natural language interface – KGQA can answer questions efficiently only when valid SPARQL queries can be formed. The forma-

Model	Answers		Answers		Passages	
	Exact Match	Exact Recall	Semantic Accuracy	Semantic Recall	Semantic Accuracy	Semantic Recall
BM25-BERT (baseline)	0.013	0.153	0.113	0.143	0.333	0.253
KGQA	0.347	0.345	0.560	0.545	-	-
BERT-QA	0.040	0.157	0.227	0.217	0.393	0.311
GPT3-QA	**0.600**	0.547	0.760	0.620	0.813	0.734
KGQA + BERT-QA	0.349	0.403	0.630	0.588	0.788	0.715
KGQA + GPT3-QA	0.500	**0.628**	**0.853**	**0.680**	**0.866**	**0.784**

Table 2: Evaluation results of 'Answers' and 'Passages' predicted by Question Answering models on Exact Match, Exact Recall, Semantic Accuracy and Semantic Recall metrics. In most metrics, KGQA + GPT3-QA performs better compared to other models.

tion of the SPARQL queries relies heavily on the NL2SPARQL pipeline, which being rule-based, is prone to errors. (b) Coverage – Not every sentence can be converted to a triple format; hence the amount of information present in the Knowledge Graph is limited. For example, the question *'What discrepancy was noted due to which flight landed at La Belle Municipal Airport?'*, which expects an answer 'problem with fuel gauge' cannot be answered using our KG as there exists no valid triple in the KG containing such information.

Limitations of DLQA module: The BERT-QA system suffers from the lack of fine-tuning on the domain dataset and hence the scores for this model are very close to the baseline. We notice that GPT3-QA does not answer multi-hop questions adequately, *i.e.*, questions that can only be answered based on the information in two different paragraphs of the same document or multiple documents. For example, the question *'Which accidents involved aircraft operated by Johnny Thornley and manufactured by Subaru?'* is not answered correctly by GPT3-QA. One needs to look at two different paragraphs in a document for fetching answers, where GPT3-QA fails. However, the responses to such questions can be obtained by traversing multiple triples in the Knowledge Graph.

Strengths of the combined system (KGQA + DLQA): Our combined system overcomes the shortcomings of the individual systems. It answers both the questions listed in the above paragraphs correctly. This combination solves the coverage issue of Knowledge Graph and the lack of domain knowledge in Deep Learning. The questions which remain unanswered by the KGQA module are answered by the DLQA module and vice versa. Both the modules complement each other, and thus the system formed by their synergy achieves better accuracy than the individual components.

The resources contributed by us are publicly available on GitHub[11].

8. Conclusion and Future Work

Aircraft safety is indispensable in the aviation domain. Aircraft accidents are unfortunate, and thus the reported accidents must be studied carefully. We have successfully created an Aviation Knowledge Graph from NTSB aircraft accident reports to help experts in their study. We also provide a Knowledge Graph guided Deep Learning based Question Answering system, which outperforms the individual KGQA and DLQA systems. The dominance of the combined QA system is proved theoretically and empirically by evaluating it on our handcrafted test set.

Currently, the constructed Knowledge Graph is based on the NTSB reports. We will continue our work on Knowledge Graph construction for ASRS reports and ADs. We aim to merge all the KGs to expand our knowledge scope in aviation safety. KGs can be extensively utilized if there is a robust querying mechanism. Thus, we need to improve our NL2SPARQL mechanism so that complex Natural Language Queries can be converted correctly to SPARQL queries. We have shown that KGQA + GPT3-QA model performs best on most metrics. We plan to improve the system by devising a mechanism to re-rank the results of the combined system or by implementing a solid combination framework.

9. Bibliographical References

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

Arnold, A., Dupont, G., Furger, F., Kobus, C., and Lancelot, F. (2020). A question-answering system for aircraft pilots' documentation. *arXiv preprint arXiv:2011.13284*.

[11]`https://github.com/RajGite/KG-assisted-DL-based-QA-in-Aviation-Domain`

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Cheng, Y., Jiao, Y., Wei, W., and Wu, Z. (2019). Research on construction method of knowledge graph in the civil aviation security field. In *2019 IEEE 1st International Conference on Civil Aviation Safety and Information Technology (ICCASIT)*, pages 556–559. IEEE.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Diefenbach, D., Singh, K., and Maret, P. (2017). Wdaqua-core0: A question answering component for the research community. In *Semantic Web Evaluation Challenge*, pages 84–89. Springer.

Diefenbach, D., Both, A., Singh, K., and Maret, P. (2020). Towards a question answering system over the semantic web. *Semantic Web*, 11(3):421–439.

Frantzi, K., Ananiadou, S., and Mima, H. (2000). Automatic recognition of multi-word terms:. the c-value/nc-value method. *International journal on digital libraries*, 3(2):115–130.

Fundel, K., Küffner, R., and Zimmer, R. (2007). Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371.

Ganu, G., Elhadad, N., and Marian, A. (2009). Beyond the stars: Improving rating predictions using review text content. In *WebDB*.

He, P., Mao, Y., Chakrabarti, K., and Chen, W. (2019). X-sql: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

Juan, R. (2003). Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

Kierszbaum, S. and Lapasset, L. (2020). Applying Distilled BERT for Question Answering on ASRS Reports. In *NTCA 2020 New Trends in Civil Aviation*, NTCA 2020 New Trends in Civil Aviation, pages 33–38, Prague, Czech Republic, November. IEEE.

Liang, S., Stockinger, K., de Farias, T. M., Anisimova, M., and Gil, M. (2021). Querying knowledge graphs in natural language. *Journal of big Data*, 8(1):1–23.

Lu, S.-Y. and Fu, K. S. (1978). A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(5):381–389.

Musen, M. A. (2015). The protégé project: a look back and a look forward. volume 1, pages 4–12.

Nadeau, D. and Sekine, S. (2007). A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad.

Reimers, N. and Gurevych, I. (2019a). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Reimers, N. and Gurevych, I. (2019b). Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11.

Robertson, S. and Zaragoza, H. (2009). *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

Sang, E. F. and De Meulder, F. (2003). Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Singh, K., Radhakrishna, A. S., Both, A., Shekarpour, S., Lytra, I., Usbeck, R., Vyas, A., Khikmatullaev, A., Punjani, D., Lange, C., et al. (2018). Why reinvent the wheel: Let's build question answering systems together. In *Proceedings of the 2018 World Wide Web Conference*, pages 1247–1256.

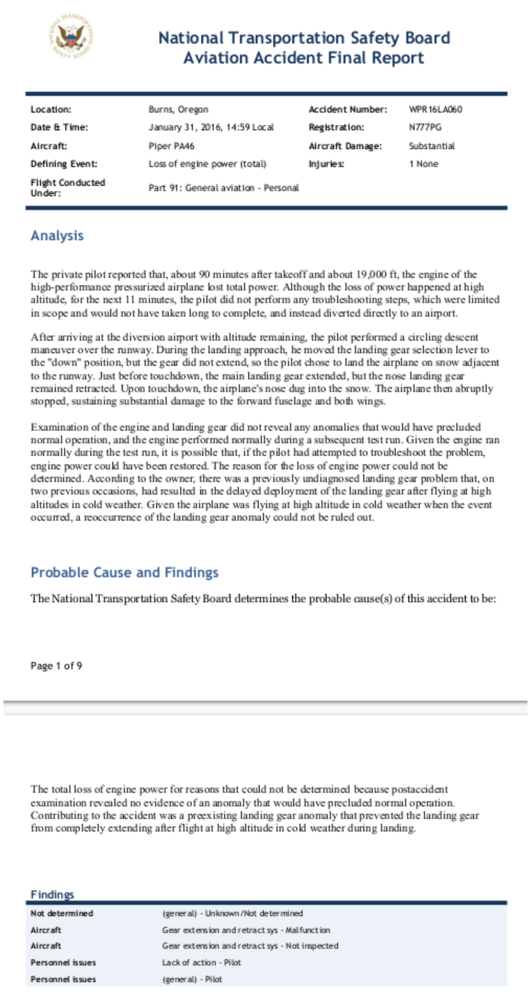
Wang, X., Yang, X., Fu, J., and Qiu, X. (2020). Research on the construction technology of knowledge graph in aviation. In *IOP Conference Series: Materials Science and Engineering*, volume 751, page 012040. IOP Publishing.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October. Association for Computational Linguistics.

Zhao, Q., Li, Q., and Wen, J. (2018). Construction and application research of knowledge graph in aviation risk field. In *MATEC Web of Conferences*, volume 151, page 05003. EDP Sciences.

Zhong, V., Xiong, C., and Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

10. Appendix



National Transportation Safety Board
Aviation Accident Final Report

Location:	Burns, Oregon	**Accident Number:**	WPR16LA060
Date & Time:	January 31, 2016, 14:59 Local	**Registration:**	N777PG
Aircraft:	Piper PA46	**Aircraft Damage:**	Substantial
Defining Event:	Loss of engine power (total)	**Injuries:**	1 None
Flight Conducted Under:	Part 91: General aviation - Personal		

Analysis

The private pilot reported that, about 90 minutes after takeoff and about 19,000 ft, the engine of the high-performance pressurized airplane lost total power. Although the loss of power happened at high altitude, for the next 11 minutes, the pilot did not perform any troubleshooting steps, which were limited in scope and would not have taken long to complete, and instead diverted directly to an airport.

After arriving at the diversion airport with altitude remaining, the pilot performed a circling descent maneuver over the runway. During the landing approach, he moved the landing gear selection lever to the "down" position, but the gear did not extend, so the pilot chose to land the airplane on snow adjacent to the runway. Just before touchdown, the main landing gear extended, but the nose landing gear remained retracted. Upon touchdown, the airplane's nose dug into the snow. The airplane then abruptly stopped, sustaining substantial damage to the forward fuselage and both wings.

Examination of the engine and landing gear did not reveal any anomalies that would have precluded normal operation, and the engine performed normally during a subsequent test run. Given the engine ran normally during the test run, it is possible that, if the pilot had attempted to troubleshoot the problem, engine power could have been restored. The reason for the loss of engine power could not be determined. According to the owner, there was a previously undiagnosed landing gear problem that, on two previous occasions, had resulted in the delayed deployment of the landing gear after flying at high altitudes in cold weather. Given the airplane was flying at high altitude in cold weather when the event occurred, a reoccurrence of the landing gear anomaly could not be ruled out.

Probable Cause and Findings

The National Transportation Safety Board determines the probable cause(s) of this accident to be:

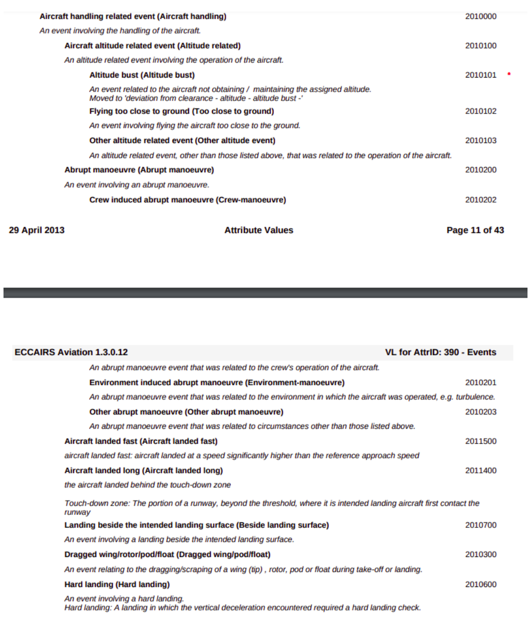
Page 1 of 9

The total loss of engine power for reasons that could not be determined because postaccident examination revealed no evidence of an anomaly that would have precluded normal operation. Contributing to the accident was a preexisting landing gear anomaly that prevented the landing gear from completely extending after flight at high altitude in cold weather during landing.

Findings

Not determined	(general) - Unknown/Not determined
Aircraft	Gear extension and retract sys - Malfunction
Aircraft	Gear extension and retract sys - Not inspected
Personnel issues	Lack of action - Pilot
Personnel issues	(general) - Pilot

Figure 3: A snippet of NTSB report



Aircraft handling related event (Aircraft handling)	2010000
An event involving the handling of the aircraft.	
Aircraft altitude related event (Altitude related)	2010100
An altitude related event involving the operation of the aircraft.	
Altitude bust (Altitude bust)	2010101 *
An event related to the aircraft not obtaining / maintaining the assigned altitude. Moved to 'deviation from clearance - altitude - altitude bust -'	
Flying too close to ground (Too close to ground)	2010102
An event involving flying the aircraft too close to the ground.	
Other altitude related event (Other altitude event)	2010103
An altitude related event, other than those listed above, that was related to the operation of the aircraft.	
Abrupt manoeuvre (Abrupt manoeuvre)	2010200
An event involving an abrupt manoeuvre.	
Crew induced abrupt manoeuvre (Crew-manoeuvre)	2010202

29 April 2013 **Attribute Values** **Page 11 of 43**

ECCAIRS Aviation 1.3.0.12 **VL for AttrID: 390 - Events**

An abrupt manoeuvre event that was related to the crew's operation of the aircraft.	
Environment induced abrupt manoeuvre (Environment-manoeuvre)	2010201
An abrupt manoeuvre event that was related to the environment in which the aircraft was operated, e.g. turbulence.	
Other abrupt manoeuvre (Other abrupt manoeuvre)	2010203
An abrupt manoeuvre event that was related to circumstances other than those listed above.	
Aircraft landed fast (Aircraft landed fast)	2011500
aircraft landed fast: aircraft landed at a speed significantly higher than the reference approach speed	
Aircraft landed long (Aircraft landed long)	2011400
the aircraft landed behind the touch-down zone	
Touch-down zone: The portion of a runway, beyond the threshold, where it is intended landing aircraft first contact the runway	
Landing beside the intended landing surface (Beside landing surface)	2010700
An event involving a landing beside the intended landing surface.	
Dragged wing/rotor/pod/float (Dragged wing/pod/float)	2010300
An event relating to the dragging/scraping of a wing (tip) , rotor, pod or float during take-off or landing.	
Hard landing (Hard landing)	2010600
An event involving a hard landing. Hard landing: A landing in which the vertical deceleration encountered required a hard landing check.	

Figure 4: A snippet of ADREP Events Taxonomy

Question: Which accidents occurred due to crosswind?
List of answers: [NYC02LA070, NYC02LA101, ...]

(a) An example depicting the output of KGQA module

Question: Which accidents occurred due to crosswind?
Answer-Passage pairs:
[NYC02LA070 - "NYC02LA070: The National Transportation Safety Board determines the probable cause(s) of this accident to be:
The student pilot's loss of aircraft control which resulted in a hard landing, and the CFI's delayed remedial action. Factors related to the accident were the fatigue cracks which initiated at the attachment hole on the landing gear strut, and the crosswind conditions.",

NYC02LA101 - "NYC02LA101: The National Transportation Safety Board determines the probable cause(s) of this accident to be:
The pilot's improper flare and his improper recovery from a bounced landing, which resulted in
a hard landing. A factor in this accident was the crosswind condition.",

...
...

]

(b) An example depicting the output of DLQA module when the underlying model is BERT-QA

Question: Which accidents occurred due to crosswind?
Answer: NYC02LA070, NYC02LA101, ...
Passages:
["NYC02LA070: The National Transportation Safety Board determines the probable cause(s) of this accident to be:
The student pilot's loss of aircraft control which resulted in a hard landing, and the CFI's delayed remedial action. Factors related to the accident were the fatigue cracks which initiated at the attachment hole on the landing gear strut, and the crosswind conditions.",

"NYC02LA101: The National Transportation Safety Board determines the probable cause(s) of this accident to be:
The pilot's improper flare and his improper recovery from a bounced landing, which resulted in
a hard landing. A factor in this accident was the crosswind condition.",

...
...

]

(c) An example depicting the output of DLQA module when the underlying model is GPT3-QA

Figure 5: Examples defining QA task

[{**Question** : Which accidents are caused due to pilot's failure to maintain control leading to collision?}, {**Answers** : SEA02FA036 , FTW02LA170 }, {**Passages** : {*SEA02FA036*: The pilot's failure to maintain aircraft control during takeoff.}, {*FTW02LA170*: Probable Cause and Findings: The National Transportation Safety Board determines the probable cause(s) of this accident to be: the pilot's failure maintain directional control of the airplane during a touch and go landing which resulted in a collision with trees. A contributing factor was the gusting wind conditions}]

Figure 6: A single instance in the QA Test Set

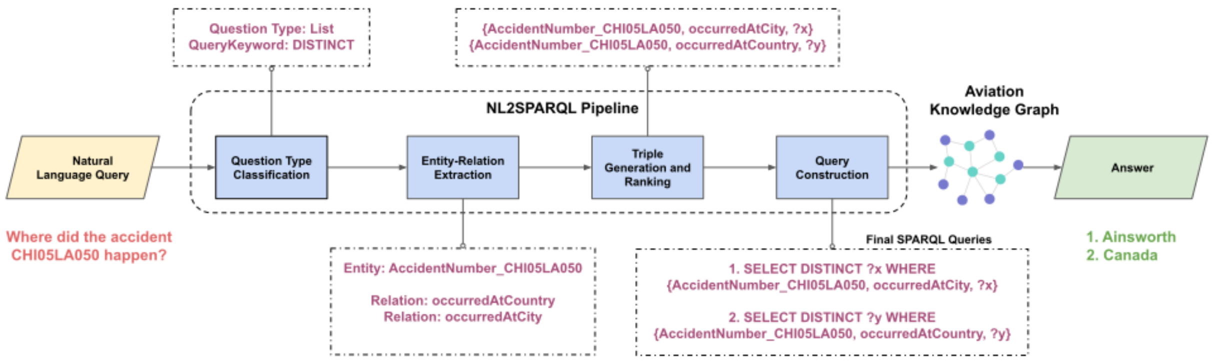
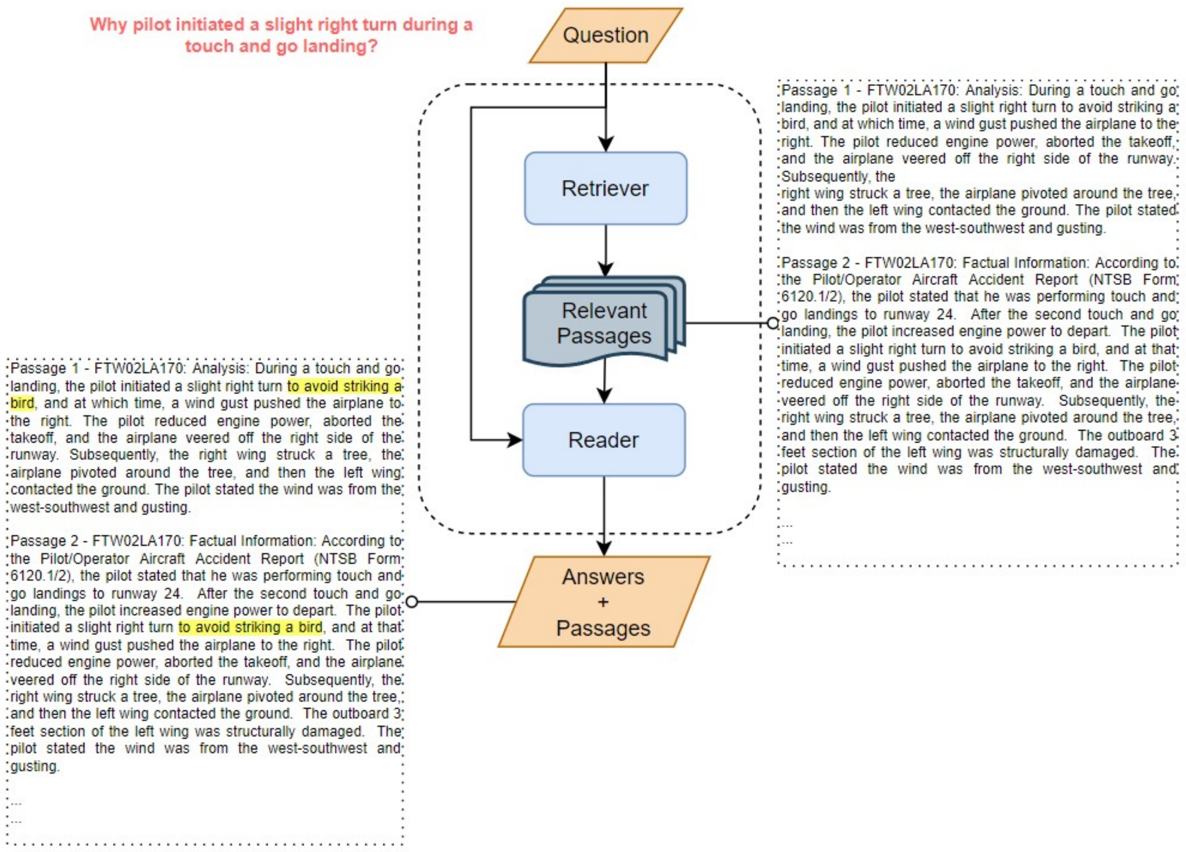

Question Type: List
QueryKeyword: DISTINCT
{AccidentNumber_CHI05LA050, occurredAtCity, ?x}
{AccidentNumber_CHI05LA050, occurredAtCountry, ?y}
NL2SPARQL Pipeline
Aviation Knowledge Graph
Natural Language Query
Question Type Classification
Entity-Relation Extraction
Triple Generation and Ranking
Query Construction
Answer
Where did the accident CHI05LA050 happen?
Entity: AccidentNumber_CHI05LA050
Relation: occurredAtCountry
Relation: occurredAtCity
Final SPARQL Queries
1. SELECT DISTINCT ?x WHERE {AccidentNumber_CHI05LA050, occurredAtCity, ?x}
2. SELECT DISTINCT ?y WHERE {AccidentNumber_CHI05LA050, occurredAtCountry, ?y}
1. Ainsworth
2. Canada

Figure 7: NL2SPARQL pipeline for KGQA system


Why pilot initiated a slight right turn during a touch and go landing?
Question
Retriever
Relevant Passages
Reader
Answers + Passages

Passage 1 - FTW02LA170: Analysis: During a touch and go landing, the pilot initiated a slight right turn to avoid striking a bird, and at which time, a wind gust pushed the airplane to the right. The pilot reduced engine power, aborted the takeoff, and the airplane veered off the right side of the runway. Subsequently, the right wing struck a tree, the airplane pivoted around the tree, and then the left wing contacted the ground. The pilot stated the wind was from the west-southwest and gusting.

Passage 2 - FTW02LA170: Factual Information: According to the Pilot/Operator Aircraft Accident Report (NTSB Form 6120.1/2), the pilot stated that he was performing touch and go landings to runway 24. After the second touch and go landing, the pilot increased engine power to depart. The pilot initiated a slight right turn to avoid striking a bird, and at that time, a wind gust pushed the airplane to the right. The pilot reduced engine power, aborted the takeoff, and the airplane veered off the right side of the runway. Subsequently, the right wing struck a tree, the airplane pivoted around the tree, and then the left wing contacted the ground. The outboard 3 feet section of the left wing was structurally damaged. The pilot stated the wind was from the west-southwest and gusting.

...
...

Passage 1 - FTW02LA170: Analysis: During a touch and go landing, the pilot initiated a slight right turn to avoid striking a bird, and at which time, a wind gust pushed the airplane to the right. The pilot reduced engine power, aborted the takeoff, and the airplane veered off the right side of the runway. Subsequently, the right wing struck a tree, the airplane pivoted around the tree, and then the left wing contacted the ground. The pilot stated the wind was from the west-southwest and gusting.

Passage 2 - FTW02LA170: Factual Information: According to the Pilot/Operator Aircraft Accident Report (NTSB Form 6120.1/2), the pilot stated that he was performing touch and go landings to runway 24. After the second touch and go landing, the pilot increased engine power to depart. The pilot initiated a slight right turn to avoid striking a bird, and at that time, a wind gust pushed the airplane to the right. The pilot reduced engine power, aborted the takeoff, and the airplane veered off the right side of the runway. Subsequently, the right wing struck a tree, the airplane pivoted around the tree, and then the left wing contacted the ground. The outboard 3 feet section of the left wing was structurally damaged. The pilot stated the wind was from the west-southwest and gusting.

...
...

Figure 8: Retriever-Reader pipeline for DLQA system. The highlighted text in the passage is the answer extracted by the DLQA system.