applied sciences

Article

Towards Domain-Specific Knowledge Graph Construction for Flight Control Aided Maintenance

Chuanyou Li [1,2,*], Xinhang Yang [3], Shance Luo [3], Mingzhe Song [4] and Wei Li [5]

[1] School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
[2] MOE Key Laboratory of Computer Network and Information Integration, Southeast University, Nanjing 211189, China
[3] Southeast University—Monash University Joint Graduate School, Soochow 215123, China
[4] Advanced Technology Research Institute, Beijing Institute of Technology, Jinan 250300, China
[5] China Southern Airlines, Beijing 102604, China
* Correspondence: cyli@seu.edu.cn

Abstract: Flight control is a key system of modern aircraft. During each flight, pilots use flight control to control the forces of flight and also the aircraft's direction and attitude. Whether flight control can work properly is closely related to safety such that daily maintenance is an essential task of airlines. Flight control maintenance heavily relies on expert knowledge. To facilitate knowledge achievement, aircraft manufacturers and airlines normally provide structural manuals for consulting. On the other hand, computer-aided maintenance systems are adopted for improving daily maintenance efficiency. However, we find that grass-roots engineers of airlines still inevitably consult unstructured technical manuals from time to time, for example, when meeting an unusual problem or an unfamiliar type of aircraft. Achieving effective knowledge from unstructured data is inefficient and inconvenient. Aiming at the problem, we propose a knowledge-graph-based maintenance prototype system as a complementary solution. The knowledge graph we built is dedicated for unstructured manuals referring to flight control. We first build ontology to represent key concepts and relation types and then perform entity-relation extraction adopting a pipeline paradigm with natural language processing techniques. To fully utilize domain-specific features, we present a hybrid method consisting of dedicated rules and a machine learning model for entity recognition. As for relation extraction, we leverage a two-stage Bi-LSTM (bi-directional long short-term memory networks) based method to improve the extraction precision by solving a sample imbalanced problem. We conduct comprehensive experiments to study the technical feasibility on real manuals from airlines. The average precision of entity recognition reaches 85%, and the average precision of relation extraction comes to 61%. Finally, we design a flight control maintenance prototype system based on the knowledge graph constructed and a graph database Neo4j. The prototype system takes alarm messages represented in natural language as the input and returns maintenance suggestions to serve grass-roots engineers.

Keywords: knowledge graph construction; named entity recognition; relation extraction; flight control system; aided maintenance system

check for updates

Citation: Li, C.; Yang, X.; Luo, S.; Song, M.; Li, W. Towards Domain-Specific Knowledge Graph Construction for Flight Control Aided Maintenance. *Appl. Sci.* **2022**, *12*, 12736. https://doi.org/10.3390/app122412736

Academic Editor: Valentino Santucci

Received: 16 November 2022
Accepted: 8 December 2022
Published: 12 December 2022

1. Introduction

In modern aircraft, flight control is the most critical system that must perform every flight mission safely. Generally, a flight control system is composed of elevator control, rudder control, and aileron control. During the lifetime of any aircraft, the flight control system inevitably has some sort of problem. For example, the linkages among different mechanical components could be loose due to overloading, aging, or fatigue-related problems. Maintaining the flight control system is indispensable and already is part of the routines of airlines.

Flight control system maintenance is a task heavily relying on expert experiences that are usually accumulated and recorded in documents. For example, from the perspective of the manufacturer of aircraft carriers, manuals such as a TSM (trouble-shooting manual) are provided to guide daily maintenance and fault diagnosis. On the other hand, airline companies also keep massive logs of fault diagnosis for future maintenance and staff training. According to our investigation, a TSM and logs of fault diagnosis are the basis of daily maintenance. A common feature of them is that they are structural and can be easily stored in a relational database. When meeting a trouble code, grass-roots engineers are capable of consulting the database to obtain effective information for fault localization.

As new aircraft are introduced, the complexity of flight control system grows. It is more and more usual that grass-roots engineers cannot quickly diagnose a problem based only on structural data. To aid engineers, computer-assisted diagnostic systems have appeared [1–6]. These systems rely on building a fault diagnosis model either by expert knowledge or massive historical records. However, when confronted with a new aircraft type or unusual problems, model-based methods are often inadequate, and engineers need to consult a set of unstructured data such as technical training manuals, flight control system instructions, etc.to localize the faults. Consulting unstructured data is inefficient and inconvenient. Aiming at the problem, we would like to study a knowledge-graph-based method, where effective information is first extracted to build a semantic knowledge base, and then an aided maintenance prototype system is built based on the knowledge extracted.

A knowledge graph built on open-source information, e.g., Wikipedia, is common. A flight control system is domain-specific such that existing open-source knowledge graphs are not adaptive. Until now, only a few studies on knowledge graphs have examined flight control [7,8]. However, they mainly focused on knowledge referring to faults (e.g., TSM) but did not draw enough attention to knowledge in other important manuals written with unstructured text. Combining with the current flight control system maintenance, we suggest that a system that can quickly provide knowledge from unstructured manuals is still missing. Hence, we will design a knowledge-graph-based system to fix the gap.

Building a knowledge graph in a new area is systematic and consists of both manual and automation work. The system overview is depicted in Figure 1. The target data are unstructured manuals referring to flight control. After format conversion and de-noising, we analyze the manuals to understand the semantics and build ontology to represent key concepts and relation types. Next, we label named entities and relations in each sentence in order to perform automatic knowledge extraction by natural language processing techniques. In our case, knowledge extraction is conducted by two sequential tasks: NER (named entity recognition) and RE (relation extraction). For named entity recognition, we present a hybrid method that combines a rule-based-algorithm and a machine-learning-based algorithm. Note that entity recognition is usually difficult to be performed by defining rules. Fortunately, flight control systems are domain-specific, and we found that some entities always appear regularly which gives us a good opportunity to recognize them by defining specific rules. For most entities, we use a machine-learning-based algorithm consisting of a Transformer encoder and a CRF (conditional random field) layer to perform recognition. In addition, to leverage features of flight control systems, we not only use the model word2vec but also POS (part of speech), domain-specific phrases, and high-frequency word information to obtain the embedding of each token. As for the relation extraction, we perform a two-stage Bi-LSTM (bi-directional long short-term memory networks) based method to relieve the sample imbalanced problem. During the first stage, a binary classification is settled that determines whether there exists a relation between a given pair of entities. After that, the second stage is carried out to differentiate the relation type. Based on the entities and relations extracted, a set of triples are achieved, and then a knowledge graph is constructed. Finally, a knowledge-graph-based prototype system is implemented on Neo4j to return possible fault locations.

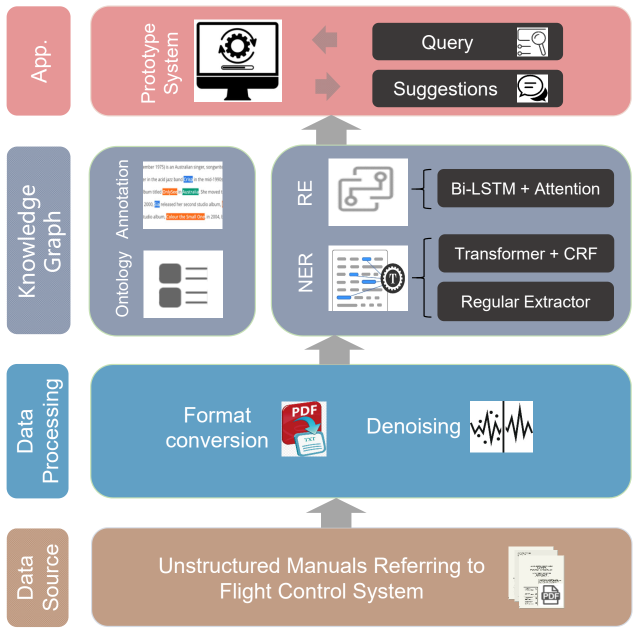


Figure 1. System overview.

The main contributions of this paper are summarized in the following:

1. We construct a knowledge graph via natural language processing techniques aiming at extracting effective knowledge from unstructured manuals referring to flight control systems.
2. To facilitate daily maintenance of flight control, an aided maintenance prototype system is implemented based on the knowledge graph we constructed. The system takes fault alarms as inputs and outputs suggestions for faults localization.
3. We set up comprehensive experiments to verify the proposed methodology. The precision of entity recognition and relation extraction, respectively, comes to 0.85 and 0.61, ensuring the availability of our prototype system.

The rest of this paper is organized as follows. Section 2 sketches out the related work, and Section 3 briefly introduces data pre-processing. In Section 4, we introduce the process of knowledge graph construction for flight control from ontology building to entity-relation extraction. Section 5 sets up experiments to evaluate the performance of entity recognition and relation extraction. Ablation experiments are also conducted to evaluate the effectiveness of domain-specific information used for token embedding. Section 6 briefly introduces a maintenance prototype built upon our knowledge graph. Finally, Section 7 brings concluding remarks and future work.

2. Related Work

In this paper, we aim at constructing a domain-specific knowledge graph for flight-control-aided maintenance. In the following text, we summarize techniques of maintenance and fault diagnosis in aircraft systems and introduce techniques referring to knowledge graph construction.

2.1. Fault Diagnosis in Aircraft Systems

Fault diagnosis methods in aircraft systems generally fall in three categories: (1) model-based method; (2) signal-based method; and (3) data-driven-based method. In model-based methods, fault diagnosis models are usually obtained by physical principles and systems identifications. Once a model is achieved, fault identification or a detection algorithm is designed to monitor the consistency between the measured outputs and the model outputs [1,2]. Signal-based methods utilize measured signals rather than input–output models. Faults are reflected in the measured signals. By extracting the features of measured signals, a fault diagnosis algorithm could be designed according to the symptoms observed and prior experiences [3,4]. Recently, data-driven methods have received a lot of attention [5,6]. This kind of method relies on historical data rather than establishing models or signal patterns. Generally, fault diagnosis is considered as a classification problem handled by deep neural networks using labeled or unlabeled data.

The method leveraged in our work is also data driven. Different from [5,6], we first build a knowledge graph from unstructured manuals and then design a maintenance prototype system to locate possible faults. We noticed that a few studies are conducted by the same knowledge-graph-based methodology [7,8]. However, they mainly focus on structural data such as TSM (trouble-shooting manual) and maintenance records to construct a knowledge graph. According to our investigation, consulting TSM or maintenance records is convenient in the current digital system. However, when confronted with a new aircraft type or unusual problems, it is inefficient to consult other unstructured manuals. Our work pays more attention to this gap where effective knowledge is extracted from unstructured data to support a better query.

2.2. Knowledge Graph Construction

A knowledge graph is a structured semantic knowledge base that can symbolically describe concepts and their relations. The basic unit of a knowledge graph is presented in the form of a triple ⟨head, relation, tail⟩, where head and tail are two types of entities that connect to each other through the relation. A knowledge graph is crucial in modern intelligent systems, such as recommendation [9,10], semantic searching [11,12], and QA (question-answering) [13,14]. By integrating fine-grained knowledge, these intelligent systems could feed back more accuracy and diversity results to human users. In the following, we focus on knowledge graph construction from the perspective of natural language processing where ontology building and entity-relation extraction are summarized.

2.2.1. Ontology Building

There is no unified methodology for ontology building. The three common methods are, respectively, the skeletal method [15], the seven-step method [16], and the cyclic acquisition method [17]. The skeletal method has four main stages: (1) identify purpose; (2) building the ontology; (3) evaluation; and (4) documentation. In addition, it is necessary to clarify the principles and guidelines for each stage. Compared with the skeletal method, the seven-step method is more delicate. It first specifies concepts and then adds attributes and relations. The seven steps are, respectively, (1) determine the domain and scope of the ontology; (2) consider reusing existing ontology; (3) enumerate important terms in the ontology; (4) define the classes and the class hierarchy; (5) define the properties of each class; (6) define the facets; and (7) create instances. The cyclic acquisition method is more adaptive for domain-specific ontology building. It first specifies a generic core ontology used as a top level structure and then acquires domain-specific concepts by a dictionary containing important corporate terms in natural language. Next, domain-unspecific concepts are removed by using a domain-specific and a general corpus of texts. Finally, relations between concepts are learned. The resulting domain-related ontology can be evaluated to decide whether it is necessary to repeat the above process to do further corrections or optimizations.

A flight control system is domain-specific. In this paper, we comprehensively leverage the ideas from the seven-step method and the cyclic acquisition method to construct ontology of the vertical area and then perform entity and relation extraction.

2.2.2. Entity and Relation Extraction

During the past decades, many works on named entity recognition and relation extraction have been proposed. Generally, there are two paradigms, pipeline extraction and joint extraction. Pipeline extraction takes entity recognition and relation extraction as two sequential tasks. Joint extraction follows a different method in that entity-relation extraction is considered as a single task. In the following text, we briefly summarize both the pipeline and joint extraction methods.

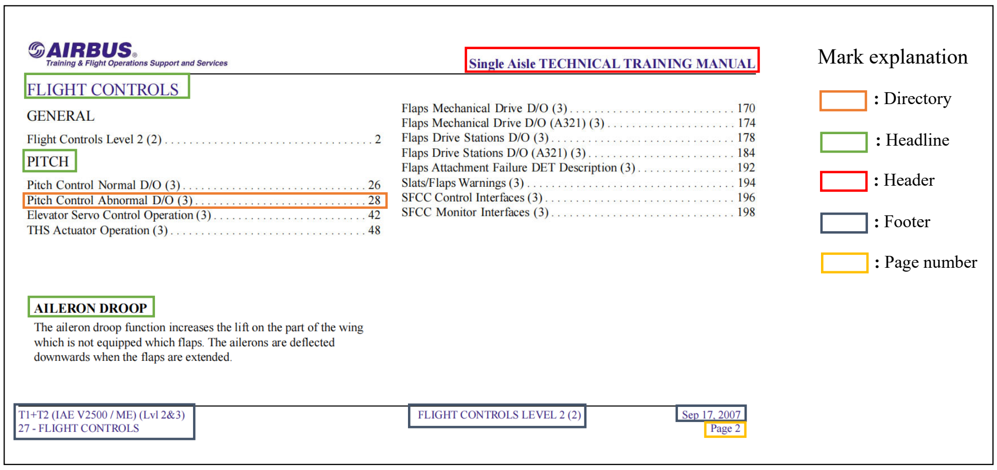
In the pipeline method, entity extraction, also known as NER (named entity recognition) is performed before relation extraction. NER usually refers to the automatic identification of named entities from text. In general, NER falls into two categories: (1) rule-based method [18]; and (2) machine-learning-based [19–25]. Rule-based methods are common in early research. However, the rule-based method is poor in scalability due to the difficulties of handling various text features. With the development of semiconductor and big data technologies, machine-learning-based methods have become dominant. This kind of method first succeeded in formal text, such as news articles [19,20], and then succeeded in informal text, such as emails, blogs [21], and tweets [22]. With the rise in deep learning techniques, NER could be modeled as a sequence annotation problem. We noted that Bi-LSTM has been widely used as a fundamental encoder for NER as it can efficiently uses both the past and future input features [23]. Recently, Transformer has been widely adopted in NLP (natural language processing) tasks because of its advantageous performance. Many researchers began to study NER by using Transformer encoder. For example, H. Yan et al. [24] proposed a model named TENER adopting a Transformer encoder to model the character-level features and word-level features. By incorporating the attention scheme, the proposed model showed its effectiveness for NER. However, the above works on NER are mainly conducted on open datasets in a general domain. Our work deals with flight control that is domain-specific. Like many other typical vertical areas [26,27], there exist many domain-specific terminologies but there is a lack of reasonable-sized and high-quality annotated datasets. In order to sufficiently leverage domain-specificity, we present a hybrid method to perform NER that combines a rule-based algorithm and a machine-learning-based algorithm. The rationality of the rule-based algorithm lies in some entities always appearing regularly. As for the machine learning algorithm, we leverage a popular methodology that is a combination of Transformer encoder and CRF (conditional random field) layer.

Relation extraction relies on extracted entities and aims at digging out the semantic connections between entities. The method of relation extraction can be divided into three categories: (1) rules- and features-based method [28,29]; (2) kernel-based method [30,31]; and (3) deep-learning-based method [32–34]. The first two categories of methods depend on manual feature extraction, and the performance strongly relies on the quality of the extracted features. Similar to the progress of NER techniques, the deep-learning-based method gradually dominates. For example, P. Zhou et al. [32] studied a Bi-LSTM-based method, T. Wu et al. [33] proposed a curriculum-meta learning method, and N. Zhang et al. [34] used knowledge graph embeddings and graph convolution networks to handle the long tail distribution problem. As for our case, sample imbalance is prominent. We leverage a two-stage Bi-LSTM-based model to relieve the problem. The first stage is a binary classification that picks out pairs of entities having a kind of relation. The second stage takes charge of distinguishing relation types.

Generally, pipeline extraction has two drawbacks. One is that the potential connection between entity and relation cannot be captured, and the other one is accumulative error. Aiming at the two drawbacks, some recent research studied joint extraction [35–38]. In this paper, we do not yet follow joint extraction. As we are in front of a vertical area with limited corpus, starting with the pipeline paradigm is easier for understanding the semantic features and on the other hand more natural for leveraging manually extracted features, e.g., defining specific rules for part of entity recognition. In addition, we also noted that there exist a set of generative methods proposed for knowledge graph construction. As our method is discriminative, we will not elaborate on them. More details referring to the generative methods can be found in [39].

3. Data Pre-Processing

Our input data are unstructured and formatted as PDFs. There are not only textual descriptions but also schematic diagrams. The information we take as input is pure text such that we choose to use the toolkit PDFMiner [40] for text parsing and TXT format conversion. Note that the PDFMiner also extracts irrelevant text such as directory, headings, headers, footers, and page numbers. We take them as noise. We extract the features of useless noise in the following. A visualized example on noise text is given in Figure 2. As long as a text line meets one of the following features, it will be removed.


AIRBUS
Training & Flight Operations Support and Services

Single Aisle TECHNICAL TRAINING MANUAL

FLIGHT CONTROLS

GENERAL

Flight Controls Level 2 (2) 2

PITCH

Pitch Control Normal D/O (3) 26
Pitch Control Abnormal D/O (3) 28
Elevator Servo Control Operation (3) 42
THS Actuator Operation (3) 48

Flaps Mechanical Drive D/O (3) 170
Flaps Mechanical Drive D/O (A321) (3) 174
Flaps Drive Stations D/O (3) 178
Flaps Drive Stations D/O (A321) (3) 184
Flaps Attachment Failure DET Description (3) 192
Slats/Flaps Warnings (3) 194
SFCC Control Interfaces (3) 196
SFCC Monitor Interfaces (3) 198

AILERON DROOP

The aileron droop function increases the lift on the part of the wing which is not equipped which flaps. The ailerons are deflected downwards when the flaps are extended.

T1+T2 (IAE V2500 / ME) (Lvl 2&3)
27 - FLIGHT CONTROLS

FLIGHT CONTROLS LEVEL 2 (2)

Sep 17, 2007
Page 2

Mark explanation

🟧 : Directory
🟩 : Headline
🟥 : Header
🟦 : Footer
🟨 : Page number

Figure 2. An example of texts that are considered noise.

1. Directory: the lines containing a sequence of ellipsis;
2. Headline: the lines only including words with uppercase letters;
3. Header: “Single Aisle TECHNICAL TRAINING MANUAL" constitutes the header appearing in each page;
4. Footer: “T1 + T2 (IAE V2500) (Lvl 2&3)”, “27—FLIGHT CONTROLS” and “Feb 01, 2011” as part of the footers in each page. There also exists a line consisting of words with uppercase letters;
5. Page number: the word “page” following by an integer number.

With the features extracted, the process of de-noising is completed by matching specific rules. In the example of Figure 2, tests in the colored boxes will be suppressed by de-noising. After de-noising, the remaining texts are separated into a sequence of sentences according to where a full stop appears.

4. Knowledge Graph Construction for Flight Control System

4.1. Ontology Building

The ontology development in our work is carried out in several rounds for corrections and optimization. In each round, the development is referred to the method proposed by Natalya F. Noy et al. [16].

Figure 3 summarizes the ontology we built including entity types and semantic relations. The whole flight control system includes five kinds of entities that are, respectively, *Equipment*, *Computer*, *System*, *Power*, and *Function* (Figure 3a). The *Equipment* class represents physical units used in the system, such as “elevator”, “rudder”, “flap”, etc. The *Computer* class includes computers, such as “slat flap control computer”, “elevator aileron computer”, “sec”, etc., leveraged in flight control for calculations and issuing commands. The entity class *System* represents different kinds of subsystems, such as “centralized fault display system”, “mechanical trim system”, and “efcs”. The *Power* class represents power sources, e.g., “electrical motor”, “hydraulic actuator”, et al. Finally, the *Function* class represents various functions that can be achieved in the flight control system, e.g., “turn coordination”, “yaw damping”, “aileron droop”, “pitch control”, “roll control”, "yaw control", etc. Based on the five entity classes, we define relations according to the semantic between different entities. Figure 3b leverages directed lines to depict all possible relations appeared between a pair of entity classes. Relations defined are, respectively, *consistOf*, *control*, *achieve*, *sendMsgTo*, *connectTo*, *locateOn*, *drive*, and *acronym*. *ConsistOf* expresses that one unit is composed of several other units. *Control* represents one unit controls multiple other units; for example, a dedicated computer controls some mechanical components. *Achieve* is defined to express one component achieving a kind of function. *SendMsgTo* defines one component sending commands or messages to other components. *LocatedOn* reflects on some kind of positional relationship. *Drive* represents power sources. Finally, *acronym* is used to express an entity's acronym.

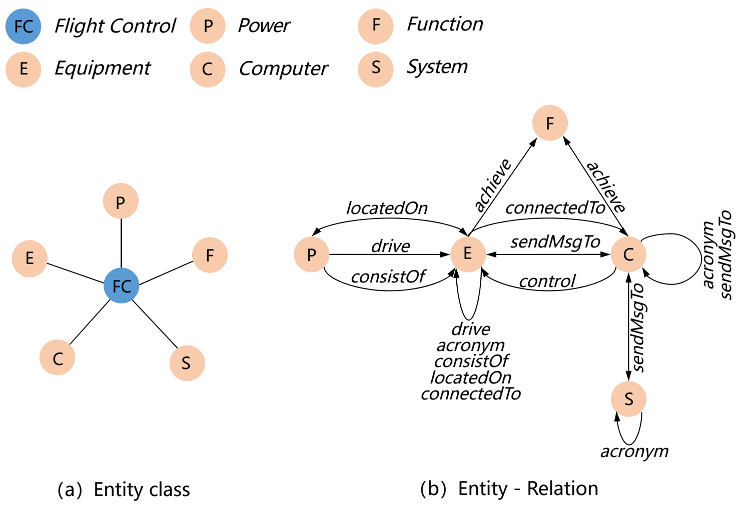


Figure 3. Overview of ontology.

4.2. Annotation

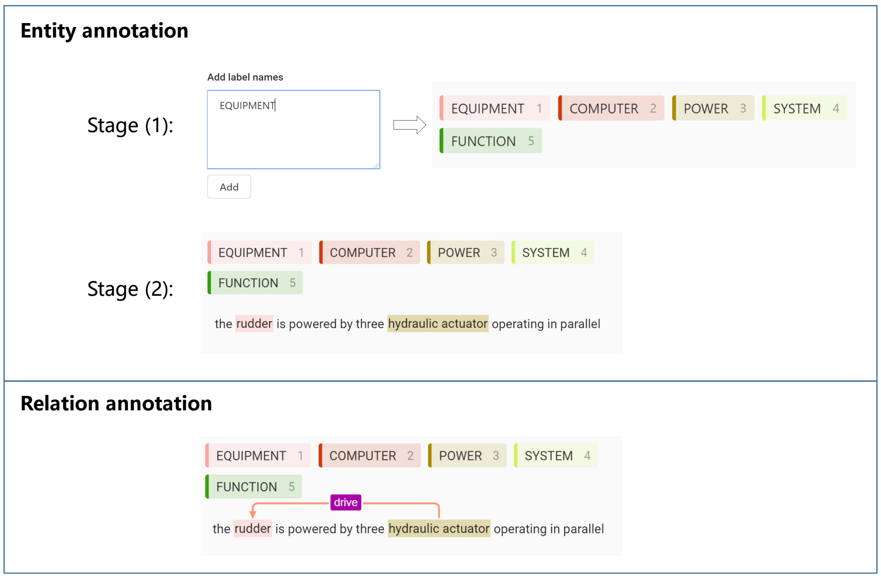
Ground truth is achieved by data annotation that is necessary for machine learning model training and verification. According to our knowledge, there is no known open labeled dataset of a flight control system especially covering those unstructured manuals. Hence, we conducted data annotation. We leverage an open-source toolkit Label-studio [41] to label the entities and relations according to the achieved ontology.

Labeling entities has two stages: (1) add each label name (representing every entity class); and (2) assign each entity a corresponding label. Relation annotation is performed by connecting two labeled entities with a specific type. Figure 4 shows that we have five different entity labels. In the instance "the rudder is powered by hydraulic actuator operating in parallel", "rudder" and "hydraulic actuator" are labeled, respectively, by *Equipment* and *Power*, and the relation between them is annotated by *drive*. After achieving a labeled dataset, we design methods for automatic named entity recognition and relation extraction.

4.3. Named Entity Recognition and Relation Extraction

In this section, we elaborate on the way of named entity recognition and relation extraction. As a flight control system is domain-specific, few referential experiences could be referred to. We adopt the basic pipeline paradigm to reach a milestone.

By careful study of the annotated text, we note that different entity classes have different features. The entity class *Power* is special: entities of this class always appear regularly such that the appearance can be described by specific rules. Unfortunately, not all entity classes show such properties. Hence, we adopt a hybrid method that first leverages a rule-based algorithm to recognize the entities belonging to *Power* and then an algorithm to recognize the remaining entities.



the rudder is powered by three hydraulic actuator operating in parallel

Figure 4. An example of entity and relation annotation.

The rule-based entity recognition method is designed based on regular expression which is a logical formula for string operations. Regular expression uses predefined specific characters and their combinations to form a pattern string. The pattern string is indeed a filter used to retrieve or replace the text conforming to the pattern. Figure 5 is the pseudo-code of the rule-based entity recognition algorithm dedicated to the *Power* class. For each input sentence, we first obtain POS (part of speech) of each token (line 3) by NLTK (natural language toolkit) [42], a famous toolkit for natural language processing. We then leverage regular expression to identify all noun phrases (line 4). The *rule* appears in line 4 and is defined to be $rule$ = <NN.* | JJ>*<NN.*> $\vee$ <JJ>*<CC><JJ>*<NN.*>, where "NN" means a noun, "." represents any character except $\backslash n$ and $\backslash r$, "*" means the token or the token sequence in <> before "*" can appear multiple times, " | " expresses logic or, "JJ" means an adjective and "CC" refers to a conjunction. The first regular expression is used for the case that a sequence of nouns or a sequence of adjectives describe a key noun phrase, while the second one handles the scenario that two sequences of adjectives concatenated by a conjunction and describe a key noun phrase. The two rules are mutually exclusive, i.e., matching any one of them is enough. Finally, we apply a simple matching that ends with "motor" or "actuator" to differentiate whether there is a noun phrase belonging to the *Power* class (lines 6–9). Let us consider the sentence "One valve block is given for each hydraulic motor" as an example. The set of noun phrases Seq_{noun} returned from line 4 is {"valve block", "hydraulic motor"}. It can be seen that "hydraulic motor" ends with motor such that it is an entity of *Power*.

```
1:  Input: a set of sentences;
2:  for each sentence do
3:      $Seq \leftarrow partOfSpeech(sentence)$;
4:      $Seq_{noun} \leftarrow nounPhrase\_extraction(Seq, rule)$;
5:      for each phrase in $Seq_{noun}$ do
6:          if phrase ends by "motor" or "actuator" then
7:              phrase is an entity of Power ;
8:          end if
9:      end for
10: end for
```

Figure 5. Extract entities belonging to *Power*.

The entity classes beyond *Power* are not easy to recognize with manual-defined rules, so we propose a machine-learning-based algorithm to handle them. Entity recognition is now modeled as a sequence annotation problem. Let us use a set $S = \{w_1, w_2, w_3, \ldots, w_n\}$ to express a sentence, where w_i is a token (word). Our target is to build a model f such that $f(S) = y_{label}$, where $f(S)$ is the annotation sequence output by the model f and y_{label} is the real annotation sequence.

The entity recognition model f we designed is based on a popular strategy combining Transformer encoder with a CRF (conditional random field) layer [24,43]. Figure 6 provides an overview of the recognition model. The Transformer encoder includes a position encoding layer and an encoder that mainly consists of multi-head self-attention and a feed-forward fully connected layer. Meanwhile, residual connection and normalization are used to connect the outputs of different hidden layers. When achieving output from the Transformer encoder, we leverage a linear and CRF layer to output the sequence of predicated labels.

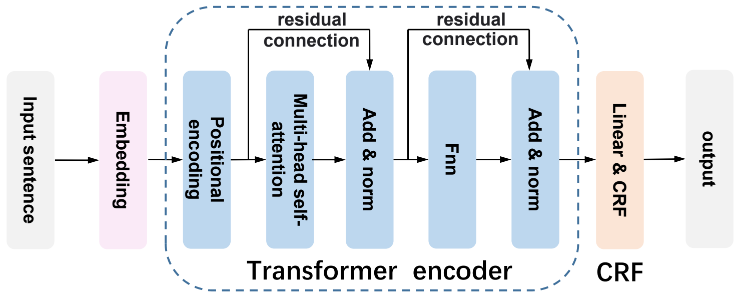


Figure 6. Named entity recognition.

For each sentence, Transformer takes its embedding as the input, which is a tensor consisting of the embedding of every token. To combine the character of the flight control system, we customize word embedding delicately by adopting information from POS, domain-specific phrase, and whether it is a high-frequency word. Specifically, there are five steps:

Step (1): The model word2vec [44] is used to achieve an embedding for every token $w_i \in S$. This process is represented by Equation (1), where $WORD_i$ is the embedding obtained.

$$WORD_i = embedding(w_i) \tag{1}$$

Step (2): As entities are all noun phrases, we take POS into the embedding. For a sentence $S = \{w_1, w_2, \ldots, w_n\}$, we use the toolkit NLTK to obtain a corresponding sequence of POS: $S_{pos} = \{pos_1, pos_2, \ldots, pos_n\}$. Embedding related to POS is represented by Equation (2), where POS_i is the output embedding.

$$POS_i = embedding_{pos}(pos_i) \tag{2}$$

Step (3): We consider domain-specific phrases and add such information into embedding. Let $S_{ph} = \{ph_1, ph_2, \ldots, ph_m\}$ include the domain-specific phrases in the dictionary of terms. For a token $w_i \in S$, if w_i matches one domain-specific phrase ph_j, then $PHRASE_j$ obtained from Equation (3) is the embedding of the domain-specific phrase. If w_i is not a domain-specific phrase, we leverage a default embedding.

$$PHRASE_i = embedding_{ph}(ph_i) \tag{3}$$

Step (4): We leverage the information of whether w_i is a high-frequency word. If w_i appears with high frequency, we denote it as "True". Otherwise, we denote it as "False". For a sentence S, we can obtain a corresponding sequence $\{hf_1, hf_2, \ldots, hf_n\}$, where each $hf_i \in \{True, False\}$. For each token w_i, HF_i achieved from Equation (4) is the embedding referring to the information of high frequency

$$HF_i = embedding_{hf}(hf_i) \tag{4}$$

Step (5): Finally, all the partial embedding vectors are concatenated together as a complete one for the word w_i:

$$total\_embedding_i = [WORD_i; POS_i; PHRASE_i; HF_i] \tag{5}$$

For a sentence S, embedding of each token is organized as a tensor which is the input of the Transformer encoder. The linear and CRF layer is used for label predication. The linear operation is for dimension transformation whose output is a sequence of vectors $\{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n\}$ corresponding to the n tokens of sentence S. CRF takes charges of scoring each candidate label sequence. For example, consider $\{\ell_0, \ell_1, \ell_2, \ldots, \ell_n, \ell_{n+1}\}$ is a candidate label sequence, where ℓ_0 and ℓ_{n+1} are two auxiliary labels representing, respectively, the starting and ending tags of the sentence. The score of a candidate label sequence is obtained by Equation (6), where P_{i,ℓ_i} is the probability that w_i's label is ℓ_i, and T is a probability transition matrix with dimension $(n+2) \times (n+2)$.

$$Score(\ell_1, \ell_2, \ldots, \ell_n) = \sum_{i=1}^{n} P_{i,\ell_i} + \sum_{i=0}^{n} T_{\ell_i,\ell_{i+1}} \tag{6}$$

Finally, a Softmax of all possible label sequences yields the probability (Equation (7)). The candidate label sequence with the highest probability is selected as the output.

$$Pr(\ell_1, \ell_2, \ldots, \ell_n | S) = \frac{e^{Score(\ell_1, \ell_2, \ldots, \ell_n)}}{\sum_{Score(\tilde{\ell}_1, \tilde{\ell}_2, \ldots, \tilde{\ell}_n)} e^{Score(\tilde{\ell}_1, \tilde{\ell}_2, \ldots, \tilde{\ell}_n)}} \tag{7}$$

Relation extraction is the task to predict the relation between two entities. As the importance of global semantic information, we consider Bi-LSTM + self-attention mechanism [32] to perform relation extraction. The attention mechanism adopted here aims at effectively obtaining key information. The schematic diagram of relation extraction is depicted in Figure 7.

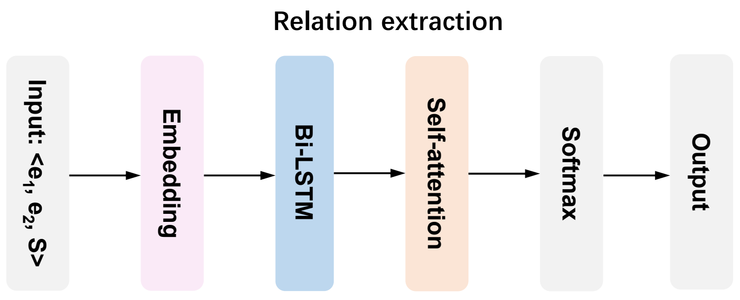


Figure 7. Relation extraction.

The input $\{e_1, e_2, S\}$ is a sentence S together with two entities e_1 and e_2. The model will predict whether in the sentence S there exists a pre-defined relation from e_1 to e_2, i.e., for such a relation e_1 and e_2 are, respectively, the head and tail entities. Different from [32], for each token w_i in S, we will obtain its embedding which is concatenated by $WORD_i$, POS_i, $HeadP_i$ and $TailP_i$. $WORD_i$ and POS_i are, respectively, from Equations (1) and (2). $HeadP_i$ and $TailP_i$ are relative position embeddings obtained by Equations (8) and (9).

$$HeadP_i = embedding_{po}(h_i) \tag{8}$$

$$TailP_i = embedding_{po}(t_i) \tag{9}$$

h_i and t_i in the above two equations are two relative positions of w_i, respectively, to the given head and tail entities. For a sentence S, we can obtain two relative position sequences $S_h = \{h_1, h_2, \ldots, h_n\}$ and $S_t = \{t_1, t_2, \ldots, t_n\}$. For example, suppose a sentence S is "the rudder does the yaw control". The head and tail entities here are, respectively, "yaw control" and "rudder". There are six tokens in S such that the position sequence relative to "yaw control" is $S_h = \{-4, -3, -2, -1, 0, 0\}$, and the position sequence relative to "rudder" is $S_t = \{-1, 0, 1, 2, 3, 4\}$. Embeddings are then put into a Bi-LSTM with the self-attention model to extract features. Finally, a Softmax layer generates the output.

Note that we do not simply use the above model to directly perform relation extraction due to us confronting a non-negligible sample imbalance problem. For example, in the sentence "The left blue and right green actuators are controlled by ELAC 1, and the other two actuators by ELAC 2", there are four named entities: "left blue and right green actuators", "ELAC 1", "actuators", and "ELAC 2". Among the 4 entities, there are 12 tuples. However, only two of them have a specific semantic relation: ⟨"ELAC 1", *control*, "left blue and right green actuators"⟩; ⟨"ELAC 2", *control*, "actuators"⟩. For the other 10 tuples, no relation exists. We take non-relation as a negative sample. Clearly, the number of negative samples dominates the dataset. In order to relieve the imbalance problem, we present a two-stage relation extraction process by leveraging the model in Figure 7. During the first stage, we train a particular relation extraction model that only takes charge of a binary classification: deciding whether there exists a defined relation between two given entities. During the section stage, we train another model to differentiate concrete relation types. Note that in the second stage it is not necessarily to consider an empty relation again such that the imbalance problem is relieved.

5. Experiments

5.1. Experimental Setup

We conducted experiments to verify the effectiveness of the entity-relation extraction methods proposed. The experiments were implemented by PyTorch 0.4.1 and CUDA 9.1. The hardware platform was composed of two Nvidia Geforce RTX 3090 (24 G) GPUs and an Intel 10980XE CPU. The testing input was "AIRBUS Training and Flight Operations Support and Services" which is an unstructured PDF and describes the composition of the flight control system. After de-noising, we obtained 1890 named entities and 4878 relations (4092 relations belonging to *Other*). Entity-relation extraction is based on sentences such that 90% of sentences were selected randomly for training, and the remaining 10% sentences were left for testing. Training epochs were set to 80, and the learning rate was 10^{-3}. We recorded P (Precision), R (Recall) and F1 score to evaluate the performance of NER and relation extraction. P, R, and F1 were calculated by Equations (10)–(12), respectively.

$$P = \frac{TP}{TP + FP} \tag{10}$$

$$R = \frac{TP}{TP + FN} \tag{11}$$

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{12}$$

TP (true positive), FP (false positive) and FN (false negative) in Equations (10) and (11) are, respectively, the number of entities (or relations) extracted correctly, false alarms, and missing entities (or relations). Table 1 provides important dimension of embeddings leveraged in both the entity recognition and relation extraction algorithms (referring to Equations (1)–(4), (8) and (9)).

Table 1. The dimension of important embeddings.

Embedding	Dimension
$WORD_i$	50
POS_i	20
$PHRASE_i$	30
HF_i	10
$HeadP_i$	10
$TailP_i$	10

5.2. Experimental Results

5.2.1. Named Entity Recognition

We first implemented NER. Remember that for a token embedding, we concatenated information from word2vec, POS, domain-specific phrase, and whether it is a high-frequency token (appearing more than 20 times). We first performed a series of ablation experiments in order to verify the effectiveness of added information. Results are given in Tables 2–5. In each table, the criteria demonstrated are, respectively, Precision, Recall, and F1.

Table 2. The results of only using embeddings from word2vec: $total\_embedding_i = [WORD_i]$.

Types of Entity	Precision	Recall	F1
Equipment	0.85	0.73	0.79
System	0.75	0.60	0.67
Computer	0.79	0.88	0.83
Function	0.62	0.67	0.64
Total	0.81	0.74	0.77

Table 3. The results of concatenating embeddings from part of speech: $total\_embedding_i = [WORD_i; POS_i]$.

Types of Entity	Precision	Recall	F1
Equipment	0.87	0.75	0.80
System	0.80	0.80	0.80
Computer	0.83	0.88	0.86
Function	0.58	0.58	0.58
Total	0.83	0.75	0.79

Table 4. The results of concatenating embeddings from domain-specific phrases: $total\_embedding_i = [WORD_i; POS_i; PHRASE_i]$.

Types of Entity	Precision	Recall	F1
Equipment	0.85	0.75	0.80
System	1.00	0.80	0.89
Computer	0.83	0.88	0.86
Function	0.59	0.83	0.69
Total	0.82	0.82	0.82

Table 5. The final results of NER: $total\_embedding_i = [WORD_i; POS_i; PHRASE_i; HF_i]$.

Types of Entity	Precision	Recall	F1
Equipment	0.87	0.80	0.84
System	1.00	0.80	0.80
Computer	0.89	0.94	0.91
Function	0.54	0.58	0.56
Total$_{ML}$	0.84	0.80	0.82
Power (by rule)	1.00	1.00	1.00
Total	0.85	0.81	0.83

From Tables 2–5, the performance of named entity recognition improves in general. Adding information of part of speech and information of domain-specific features can help the model to better distinguish entities. We provide two sentences as an example to show the progress of entity recognition.

1. If the aircraft reaches the pitch attitude protection nose up limit, then the flight control computer will override pilot demand and keep the aircraft within the safe flight limit.
2. In case of failure of both elac 1 servo control, then elac 2 does the computation and control its servo control.

In the first sentence, "flight control computer" is an entity that can be correctly recognized if integrating information from part of speech and domain-specific features. However, if only using embeddings from word2vec, the model can only recognize "computer", making a mistake on entity boundary. In the second sentence, the last two words "servo control" is an entity. Again, it cannot be recognized as an entity if only using embeddings from word2vec.

Table 5 integrates results from both the machine-learning-based algorithm and the rule-based algorithm. As the *Power* class is special, the rule-based method already covers all possible cases such that all entities of *Power* can be recognized accurately. Combining all results (the last row of Table 5), we think that the performance of named entity recognition reaches a reasonable level.

In addition, we have noticed the weakness of the current method, especially when recognizing entities of the *Function* class. Testing results on all the three criteria are not good compared with results on other entity classes. After a careful study, we find that entities belonging to *Function* are diverse from each other. Training samples are not abundant enough to extract a variety of diverse features. Hence, missing recognition and wrong entity boundaries are more prone to happen compared with recognizing entities of other classes. For example, we find that the model always recognizes "rudder deflection" rather than "rudder deflection limitation" due to both of them being a kind of function. On the other hand, we find that recognizing entities ended by "extension" is difficult for the model. For example, the model fails to recognize "ground spoiler extension". The training set does not directly include "ground spoiler extension" but has "flap extension" and "speed brake extension". However, due to the number of samples being limited, the model does not achieve enough features. To fix these drawback, on the one hand, we add more domain-specific corpus, e.g., adopting some textbook in a flight control system. On the other hand, more fine-grained techniques could be involved such as Pointer Network to handle nested entities.

5.2.2. Relation Extraction

The results of relation extraction are given, respectively, in Tables 6–9. The criteria demonstrated are still P (Precision), R (Recall), and F1 score. Let us combine the results to explain one by one.

Table 6. Binary relation extraction.

Types of Relation	Precision	Recall	F1
defined relation	0.89	0.81	0.85
Other	0.98	0.99	0.98

Table 7. Two-stage method: relation extraction results based on annotated entities.

Types of Relation	Precision	Recall	F1
ConsistOf	1.00	0.75	0.86
Control	0.80	1.00	0.92
Achieve	0.00	0.00	0.00
Acronym	0.71	1.00	0.83
Drive	0.56	1.00	0.92
SendMsgTo	1.00	0.50	0.67
ConnectedTo	1.00	1.00	1.00
LocatedOn	1.00	1.00	1.00
Average	0.86	0.77	0.81

Table 8. One-stage method: relation extraction results based on annotated entities.

Types of Relation	Precision	Recall	F1
ConsistOf	1.00	0.25	0.40
Control	1.00	1.00	1.00
Achieve	0.00	0.00	0.00
Acronym	0.71	1.00	0.83
Drive	0.80	0.67	0.73
SendMsgTo	0.75	0.75	0.75
ConnectedTo	0.67	1.00	0.80
LocatedOn	0.67	1.00	0.80
Average	0.75	0.68	0.71

Table 9. Relation extraction based on recognized entities.

Types of Relation	Precision	Recall	F1
ConsistOf	0.75	0.75	0.75
Control	0.67	0.50	0.57
Achieve	0.00	0.00	0.00
Acronym	0.43	0.60	0.50
Drive	0.75	0.50	0.60
SendMsgTo	1.00	0.25	0.40
ConnectedTo	1.00	0.50	0.67
LocatedOn	0.50	0.50	0.50
Average	0.61	0.45	0.52

Remember that we leverage a two-stage relation extraction procedure, where the first stage is a binary classification problem. We involve this stage aiming at eliminating the impact generated by excessive negative samples. Results in Table 6 show the model can well differentiate (up to 98%) the cases having no defined relation, and meanwhile it can correctly keep more than 80% defined relations. During the second stage, we train a model to distinguish different types of defined relations. Table 7 provides the results of the two-stage method. It can be seen that with prior knowledge on correct pair of entities, the model can well recognize the type of relations: the average Precision reaches 0.86 and the average F1 reaches 0.81. We also noticed that relation *Achieve* cannot be extracted by the model. The problem arises from the binary classification. At this stage model extract common features of all existence relations. Unfortunately, samples of *Achieve* are diverse and the number is limited such that the model is prone to make a mistake. To demonstrate the effectiveness of the two-stage method, we also give out the results by directly training the model (named one-stage) for comparison. The results are provided in Table 8. It can be seen that by the one-stage method, the average performance decreases by more 10 percentages.

Note that results in Table 7 are obtained based on correct entities. To obtain the final results for knowledge graph construction, it is necessary to combine with the named entity recognition. Table 9 provides the final results on relation extraction. Compared with results in Table 7, there exists an apparent performance decrease. Faults in NER directly encumber the relation extraction. As long as an entity is not correctly recognized, all the relations related to such entity cannot be correctly extracted too. To continue improving the quality of entity-relation extraction, it is necessary to add more corpus and on the other hand, to reduce the accumulation error, for example, adopting the joint extraction paradigm.

5.3. A Case Study of Knowledge Graph Construction

In this section, we provide a concrete case study to show the process of knowledge graph construction. We extract three sentences from the input document referring to a flight control system and take them as the input of the case study.

1. In flight, the elac transmits the yaw damp and turn coordination signal to the flight augmentation computer (fac);
2. tiThe side stick sends electrical order to the elevator aileron computer (elac) and spoiler elevator computer (sec);
3. Elac normally controls the elevator and trimmable horizontal stabilizer (ths).

By executing NER, we obtain three entities {"elac", "flight augmentation computer", "fac"} from the first sentence,five entities {"side stick", "elevator aileron computer", "elacs", "spoiler elevator computer", "sec"} from the second sentence, and four entities {"elac", "elevator", "trimmable horizontal stabilizer", "ths"} from the third sentence.

As a relation exists in a pair of entities, we first enumerate all permutations for a pair of entities within a sentence. For example, six entity pairs can be obtained from the first sentence, which are, respectively: ⟨"elac", "flight augmentation computer"⟩, ⟨"elac", "fac"⟩, ⟨"flight augmentation computer", "elac"⟩, ⟨"fac", "elac"⟩, ⟨"flight augmentation computer", "fac"⟩, and ⟨"fac", "flight augmentation computer"⟩. Similarly, there are 20 and 12 entity pairs that can be enumerated, respectively, from the second and the third sentences. In the first stage of relation extraction, we keep the entity pairs if there exists a defined relation. For instance, among the six entity pairs from the first sentence, ⟨"elac", "flight augmentation computer"⟩ and ⟨"flight augmentation computer", "fac"⟩ are kept. During the second stage, definite relation types are recognized, for example, ⟨"elac", *sendMsgTo*, "flight augmentation computer"⟩ and ⟨"flight augmentation computer", *acronym*, "fac"⟩. After extracting all relations from the three sentences, the corresponding partial knowledge graph (see Figure 8) is constructed based on Neo4j.

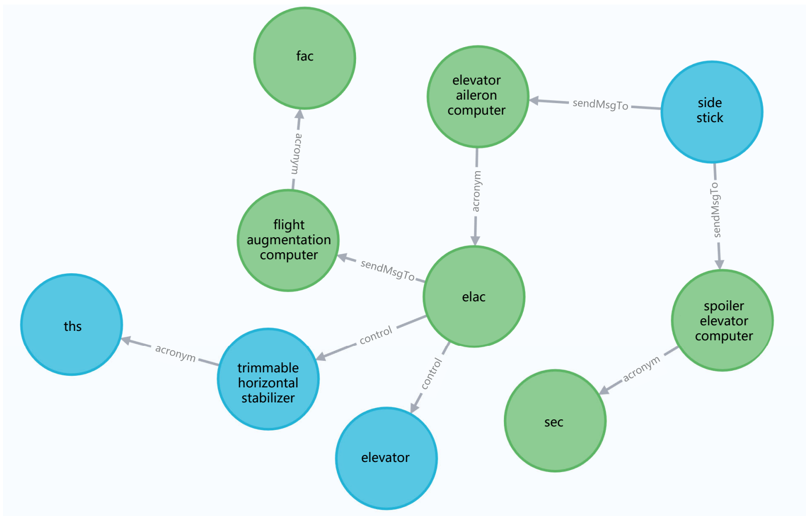

fac
elevator aileron computer
sendMsgTo
side stick
acronym
acronym
sendMsgTo
flight augmentation computer
sendMsgTo
elac
ths
acronym
control
trimmable horizontal stabilizer
control
spoiler elevator computer
acronym
sec
elevator

Figure 8. A case study of knowledge graph construction.

6. Application for Aided Maintenance of Flight Control System

After entity-relation extraction, we construct a domain-specific knowledge graph by organizing a set of triples. The knowledge graph constructed is stored in Neo4j that is a high-performance NOSQL graphical database. Different from a relational database to represent data in tables, Neo4j adopts nodes and edges to represent data, where nodes are specific entities, and edges are relations between entities. Based on Neo4j, we implement a web-based human–computer interaction system prototype aimed at aided maintenance for grass-roots engineers. The main interface of the system is given in Figure 9, where the search box takes the input of a query.

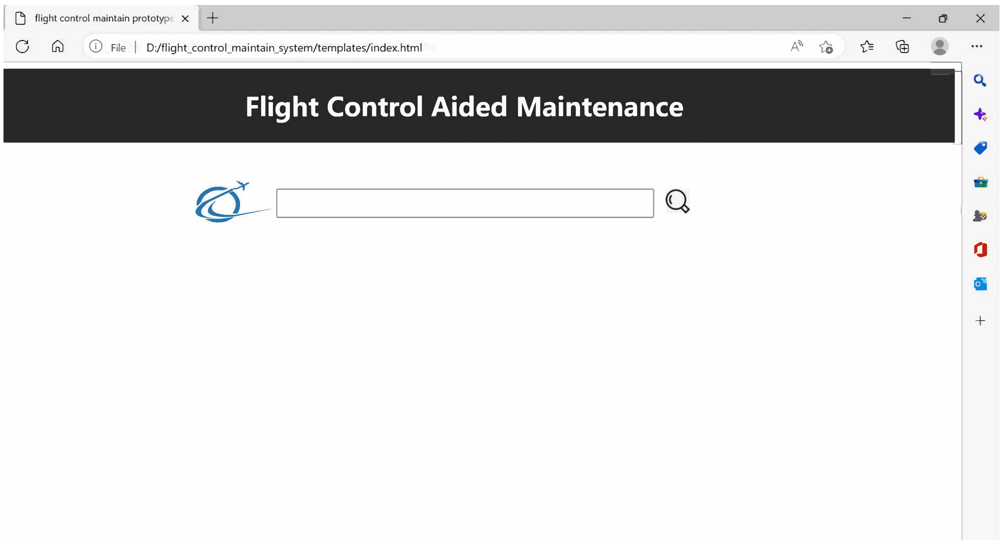
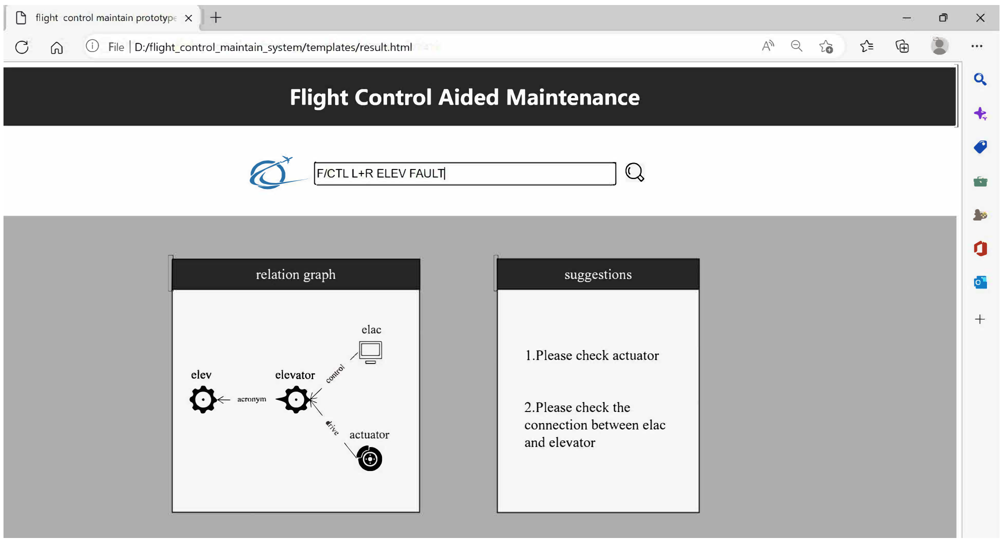

flight control maintain prototype
File | D:/flight_control_maintain_system/templates/index.html
Flight Control Aided Maintenance

Figure 9. Main interface of flight control aided maintenance system.

In a real maintenance scenario, grass-roots engineers normally start with alarm messages, e.g., “F/CTL L + R ELEV FAULT”, “F/CTL SEC 1 FAULT”, etc. According to current processing specification, engineers first consult the alarms from TSM and historical records (in relational database). If the problem is solved according to official recommended operations or accumulated fault diagnosis experiences, the maintenance is complete. When engineers need more information from unstructured data, our prototype system works. As for the above two examples, the alarm messages will be typed into the search box. Named entity recognition is then carried out to capture all entities (“ELEV” and “SEC 1”) from the alarm messages. Next, a query of “ELEV” and “SEC 1” is performed on the knowledge graph. For “ELEV”, the query first finds that it is an acronym for “elevator” such that a recursive query of “elevator” happens automatically to obtain more information.

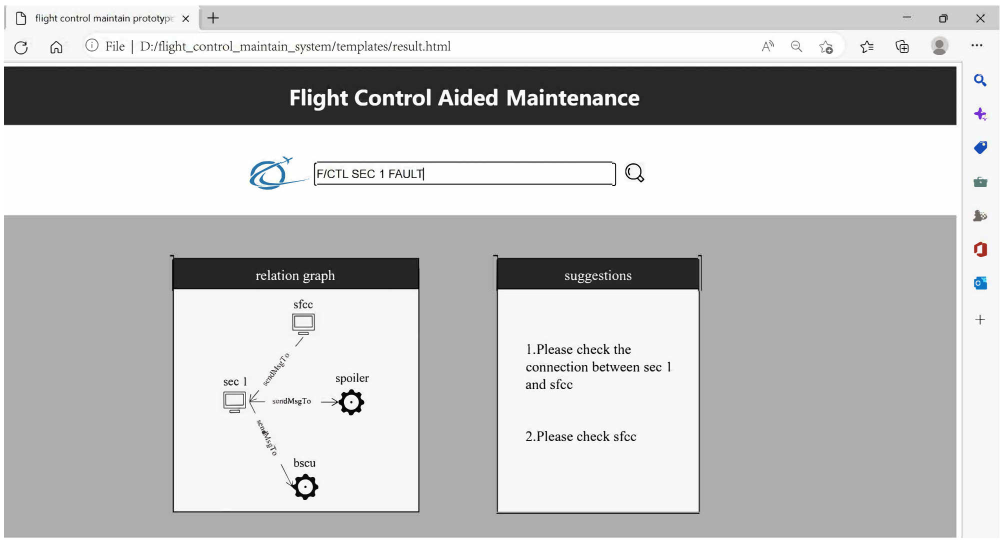
In the knowledge graph, the triples related to “elevator” are ⟨“actuator”, *drive*, “elevator”⟩, ⟨“elac”, *control*, “elevator”⟩, and the triples related to “sec 1” are ⟨“sfcc”, *sendMsgTo*, “sec 1”⟩, ⟨“sec 1”, *sendMsgTo*, “bscu”⟩, and ⟨“sec 1”, *sendMsgTo*, “spoiler”⟩. Because “actuator” drives “elevator” and “elac” controls “elevator”, it could be the case that a failure appears in “actuator” or “elac”, and it also could be the case the connection between “actuator” and “elevator” or the connection between “elac” and “elevator” suffered a failure. As for “sec 1”, it could be the case that it fails to receive or send key messages such that the system recommends to check “sfcc” and the related connection cables. Query results from our prototype system are shown, respectively, in Figures 10 and 11.


flight control maintain prototype
File | D:/flight_control_maintain_system/templates/result.html
Flight Control Aided Maintenance
F/CTL L+R ELEV FAULT
relation graph
elac
elev
elevator
acronym
control
drive
actuator
suggestions

1.Please check actuator

2.Please check the connection between elac and elevator

Figure 10. Example of querying "F/CTL L + R ELEV FAULT".


flight control maintain prototype
File | D:/flight_control_maintain_system/templates/result.html
Flight Control Aided Maintenance
F/CTL SEC 1 FAULT
relation graph
sfcc
sendMsgTo
sec 1
spoiler
sendMsgTo
sendMsgTo
bscu
suggestions

1.Please check the connection between sec 1 and sfcc

2.Please check sfcc

Figure 11. Example of querying "F/CTL SEC 1 FAULT".

7. Conclusions

In this paper, we propose a knowledge-graph-based system prototype in order to facilitate achieving knowledge from unstructured data for daily flight control maintenance. We first build ontology and then perform entity-relation extraction by leveraging natural language processing techniques. Finally, we design and implement a flight control maintenance system prototype based on the knowledge graph constructed whose duty is to return maintenance suggestions for grass-roots engineers.

Our work is a start-up of promoting flight control maintenance by leveraging information technologies. In our prototype system, a natural language-processing-based method plays a key role. However, in the vertical area of flight control, the lack of sufficient corpus restrains the performance of a machine learning model. Hence, we will improve our system from three aspects in the future. First, finding more corpus in this area, such as textbooks, research papers or even related technical web pages. Second, adopting more techniques to improve the extraction model, such as joint extraction, Pointer Network, pretrained word embedding model such as BERT (bidirectional encoder representation from Transformers), and generative methods. Finally, we will expand the system to adopt all structured data to replace the current relation-database-based system.

Author Contributions: Conceptualization, C.L., M.S. and W.L.; formal analysis, M.S. and W.L.; funding acquisition, C.L.; investigation, S.L.; methodology, X.Y., M.S. and W.L.; project administration, C.L. and M.S.; resources, W.L.; software, X.Y. and S.L.; supervision, W.L.; validation, W.L.; writing—original draft, C.L.; writing—review and editing, C.L., X.Y., S.L. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China under Grant 61902063, by the Provincial Natural Science Foundation of Jiangsu, China under Grant BK20190342, and by the Open Project Program of the State Key Laboratory of Mathematical Engineering and Advanced Computing by Grant 2020A04.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

NLP	Natural Language Processing
NER	Named Entity Recognition
RE	Relation Extraction
NLTK	Natural Language Toolkit
POS	Part of Speech
TSM	Trouble-Shooting Manual
Bi-LSTM	Bi-directional Long Short-term Memory networks
CRF	Conditional Random Field
QA	Question-Answering
P	Precision
R	Recall
TP	True Positive
FP	False Positive
FN	False Negative
BERT	Bidirectional Encoder Representation from Transformers

References

1. Goupil, P. Oscillatory failure case detection in the A380 electrical flight control system by analytical redundancy. *Control Eng. Pract.* **2010**, *18*, 1110–1119. [CrossRef]
2. Bobrinskoy, A.; Cazaurang, F.; GATTI, M.; Guerineau, O.; Bluteau, B. Model-based fault detection and isolation design for flight-critical actuators in a Harsh environment. In Proceedings of the 31st IEEE/AIAA Conference on Digital Avionics Systems (DASC), Williamsburg, VA, USA, 14–18 October 2012; pp. 1–17.
3. Lin, C.; Liu, C. Failure Detection and Adaptive Compensation for Fault Tolerable Flight Control Systems. *IEEE Trans. Ind. Inform.* **2007**, *3*, 322–331. [CrossRef]
4. Van Eykeren, L.; Chu, Q. Fault Detection and Isolation for Inertial Reference Units. In Proceedings of the AIAA Conference on Guidance, Navigation, and Control (GNC), Boston, MA, USA, 19–22 August 2013; pp. 1–10.
5. Wen, L.; Li, X.; Gao, L.; Zhang, Y. A New Convolutional Neural Network-Based Data-Driven Fault Diagnosis Method. *IEEE Trans. Ind. Electron.* **2018**, *65*, 5990–5998. [CrossRef]
6. Song, X.; Zheng, Z.; Guan, Z.; Yang, D.; Liu, R. Deep learning Fault Diagnosis in Flight Control System of Carrier-Based Aircraft. In Proceedings of the 17th IEEE International Conference on Control & Automation (ICCA), Naples, Italy, 27–30 June 2022; pp. 492–497.
7. Tang, X.; Wang, J.; Wu, C.; Hu, B.; Noman, S.M. Constructing Aircraft Fault Knowledge Graph for Intelligent Aided Diagnosis. In Proceedings of the 4th International Conference on Information Technologies and Electrical Engineering, Changde, China, 29–31 October 2021; pp. 1–4.
8. Zhang, S.; Zhang, Y.; Yang, Y.; Cheng, W.; Zhao, H.; Li, Y. Knowledge Graph Construction for Fault Diagnosis of Aircraft Environmental Control System. In Proceedings of the Global Reliability and Prognostics and Health Management (PHM-Nanjing), Nanjing, China, 15–17 October 2021; pp. 1–5.

9. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In Proceedings of the World Wide Web Conference (WWW), San Francisco, CA, USA, 13–17 May 2019; pp. 2000–2010.
10. Bellini, V.; Anelli, V.W.; Noia, T.D.; Sciascio, E.D. Auto-Encoding User Ratings via Knowledge Graphs in Recommendation Scenarios. In Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, Como, Italy, 27 August 2017; pp. 60–66.
11. Zhang, N.; Jia, Q.; Deng, S.; Chen, X.; Ye, H.; Chen, H.; Tou, H.; Huang, G.; Wang, Z.; Hua, N.; et al. AliCG: Fine-grained and Evolvable Conceptual Graph Construction for Semantic Search at Alibaba. In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, 14–18 August 2021; pp. 3895–3905.
12. Shi, Y.; Cheng, G.; Tran, T.; Kharlamov, E.; Shen, Y. Efficient Computation of Semantically Cohesive Subgraphs for Keyword-Based Knowledge Graph Exploration. In Proceedings of the Web Conference, Virtual Event, 19–23 April 2021; pp. 1410–1421.
13. Bordes, A.; Chopra, S.; Weston, J. Question Answering with Subgraph Embeddings. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; pp. 615–620.
14. He, Y.; Zhu, Z.; Zhang, Y.; Chen, Q.; Caverlee, J. Infusing Disease Knowledge into BERT for Health Question Answering, Medical Inference and Disease Name Recognition. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 4604–4614.
15. Uschold, M.; King, M. Towards a Methodology for Building Ontologies. In Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI 1995, Montreal, QC, Canada, 20–25 August 1995; pp. 1–13.
16. Noy, N.F.; McGuinness, D.L. *Ontology Development 101: A Guide to Creating Your First Ontology*; Technical Report, Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880; 2001. Available online: https://protege.stanford.edu/publications/ontology_development/ontology101.pdf (accessed on 5 June 2022).
17. Kietz, J.U.; Volz, R.; Maedche, A. Extracting a Domain-Specific Ontology from a Corporate Intranet. In Proceedings of the 2nd Workshop on Learning Language in Logic and the 4th Conference on Computational Natural Language Learning, Lisbon, Portugal, 13–14 September 2000; pp. 167–175.
18. Rau, L.F. Extracting company names from text. In Proceedings of the 7th IEEE Conference on Artificial Intelligence Application, Herndon, VA, USA, 5–8 November 1995; pp. 29–32.
19. McCallum, A.; Li, W. Early results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. In Proceedings of the 7th Conference on Natural Language Learning, CoNLL, Held in cooperation with HLT-NAACL, Edmonton, AB, Canada, 31 May–1 June 2003; pp. 188–191.
20. Etzioni, O.; Cafarella, M.J.; Downey, D.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D.S.; Yates, A. Unsupervised named-entity extraction from the Web: An experimental study. *Artif. Intell.* **2005**, *165*, 91–134. [CrossRef]
21. Wang, Y. Annotating and Recognising Named Entities in Clinical Notes. In Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP, Student Research Workshop, Singapore, 2–7 August 2009; pp. 18–26.
22. Liu, X.; Zhang, S.; Wei, F.; Zhou, M. Recognizing Named Entities in Tweets. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OG, USA, 19–24 June 2011; pp. 359–367.
23. Huang, Z.; Xu, W.; Yu, K. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv* **2015**, arXiv:1508.01991.
24. Yan, H.; Deng, B.; Li, X.; Qiu, X. TENER: Adapting Transformer Encoder for Named Entity Recognition. *arXiv* **2019**, arXiv:1911.04474.
25. Liu, Z.; Jiang, F.; Hu, Y.; Shi, C.; Fung, P. NER-BERT: A Pre-trained Model for Low-Resource Entity Tagging. *arXiv* **2021**, arXiv:2112.00405.
26. Tarcar, A.K.; Tiwari, A.; Rao, D.; Dhaimodker, V.N.; Rebelo, P.; Desai, R. Healthcare NER Models Using Language Model Pretraining. In Proceedings of the ACM WSDM Health Search and Data Mining Workshop, co-located with the 13th ACM International WSDM Conference, Houston, TX, USA, 3 February 2020; Volume 2551, pp. 12–18.
27. Syed, M.H.; Chung, S.T. MenuNER: Domain-Adapted BERT Based NER Approach for a Domain with Limited Dataset and Its Application to Food Menu Domain. *Appl. Sci.* **2021**, *11*, 6007. [CrossRef]
28. Kambhatla, N. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) on Interactive Poster and Demonstration Sessions, Barcelona, Spain, 21–26 July 2004; pp. 22–25.
29. Zhou, G.; Su, J.; Zhang, J.; Zhang, M. Exploring Various Knowledge in Relation Extraction. In Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL), Ann Arbor, MI, USA, 25–30 June 2005; pp. 427–434.
30. Culotta, A.; Sorensen, J.S. Dependency Tree Kernels for Relation Extraction. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL), Barcelona, Spain, 21–26 July 2004; pp. 423–429.
31. Bunescu, R.C.; Mooney, R.J. A Shortest Path Dependency Kernel for Relation Extraction. In Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), Vancouver, BC, Canada, 6–8 October 2005; pp. 724–731.
32. Zhou, P.; Shi, W.; Tian, J.; Qi, Z.; Li, B.; Hao, H.; Xu, B. Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany, 7–12 August 2016.

33. Wu, T.; Li, X.; Li, Y.; Haffari, G.; Qi, G.; Zhu, Y.; Xu, G. Curriculum-Meta Learning for Order-Robust Continual Relation Extraction. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, Virtual Event, 2–9 February 2021; pp. 10363–10369.
34. Zhang, N.; Deng, S.; Sun, Z.; Wang, G.; Chen, X.; Zhang, W.; Chen, H. Long-tail Relation Extraction via Knowledge Graph Embeddings and Graph Convolution Networks. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT), Minneapolis, MN, USA, 2–7 June 2019; pp. 3016–3025.
35. Miwa, M.; Bansal, M. End-to-End Relation Extraction using LSTMs on Sequences and Tree Structures. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL), Berlin, Germany, 7–12 August 2016.
36. Katiyar, A.; Cardie, C. Going out on a limb: Joint Extraction of Entity Mentions and Relations without Dependency Trees. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL), Vancouver, BC, Canada, 30 July–4 August 2017; pp. 917–928.
37. Shang, Y.; Huang, H.; Mao, X. OneRel: Joint Entity and Relation Extraction with One Module in One Step. In Proceedings of the 36th Conference on Artificial Intelligence (AAAI),Virtual Event, 22 February–1 March 2022; pp. 11285–11293.
38. Wei, Z.; Su, J.; Wang, Y.; Tian, Y.; Chang, Y. A Novel Cascade Binary Tagging Framework for Relational Triple Extraction. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ALC), Online, 5–10 July 2020; pp. 1476–1488.
39. Ye, H.; Zhang, N.; Chen, H.; Chen, H. Generative Knowledge Graph Construction: A Review. *arXiv* **2022**, arXiv:2210.12714.
40. Pdfminer. Available online: https://www.unixuser.org/~euske/python/pdfminer (accessed on 5 June 2022).
41. Label-Studio. Available online: https://labelstud.io (accessed on 15 October 2022).
42. Bird, S; Loper, E.; Klein, E. *Natural Language Processing with Python*; Technical Report; O'Reilly Media Inc.: Sebastopol, CA, USA, 2009.
43. Li, X.; Yan, H.; Qiu, X.; Huang, X. FLAT: Chinese NER Using Flat-Lattice Transformer. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL), Online, 5–10 July 2020; pp. 6836–6842.
44. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, AZ, USA, 2–4 May 2013.