aerospace

MDPI

Article

aeroBERT-Classifier: Classification of Aerospace Requirements Using BERT

Archana Tikayat Ray [1,*], Bjorn F. Cole [2], Olivia J. Pinon Fischer [1,*], Ryan T. White [3] and Dimitri N. Mavris [1]

[1] Aerospace Systems Design Laboratory, School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[2] Lockheed Martin Space, Littleton, CO 80127, USA
[3] Department of Mathematical Sciences, Florida Institute of Technology, Melbourne, FL 32901, USA
* Correspondence: atr@gatech.edu (A.T.R.); olivia.pinon@asdl.gatech.edu (O.J.P.F.)

Abstract: The system complexity that characterizes current systems warrants an integrated and comprehensive approach to system design and development. This need has brought about a paradigm shift towards Model-Based Systems Engineering (MBSE) approaches to system design and a departure from traditional document-centric methods. While MBSE shows great promise, the ambiguities and inconsistencies present in Natural Language (NL) requirements hinder their conversion to models directly. The field of Natural Language Processing (NLP) has demonstrated great potential in facilitating the conversion of NL requirements into a semi-machine-readable format that enables their standardization and use in a model-based environment. A first step towards standardizing requirements consists of classifying them according to the type (design, functional, performance, etc.) they represent. To that end, a language model capable of classifying requirements needs to be fine-tuned on labeled aerospace requirements. This paper presents an open-source, annotated aerospace requirements corpus (the first of its kind) developed for the purpose of this effort that includes three types of requirements, namely design, functional, and performance requirements. This paper further describes the use of the aforementioned corpus to fine-tune BERT to obtain the aeroBERT-Classifier: a new language model for classifying aerospace requirements into design, functional, or performance requirements. Finally, this paper provides a comparison between aeroBERT-Classifier and other text classification models such as GPT-2, Bidirectional Long Short-Term Memory (Bi-LSTM), and bart-large-mnli. In particular, it shows the superior performance of aeroBERT-Classifier on classifying aerospace requirements over existing models, and this is despite the fact that the model was fine-tuned using a small labeled dataset.

Keywords: requirements engineering; natural language processing; NLP; BERT; requirements classification; text classification

Citation: Tikayat Ray, A.; Cole, B.F.; Pinon Fischer, O.J.; White, R.T.; Mavris, D.N. aeroBERT-Classifier: Classification of Aerospace Requirements Using BERT. *Aerospace* **2023**, *10*, 279. https://doi.org/10.3390/aerospace10030279

Academic Editor: Gokhan Inalhan

Received: 3 February 2023
Revised: 9 March 2023
Accepted: 10 March 2023
Published: 11 March 2023

1. Introduction

This section provides an introduction to the field of requirements engineering and discusses its importance in the design of systems, products, and enterprises. It also emphasizes the challenges and limitations associated with the use of Natural Language (NL) for requirements elicitation. It then discusses the benefits of a shift towards model-based approaches and examines the hindrance such shift encounters due to the ambiguities inherent to NL requirements. This section finally concludes with the focus of the present research and its contributions.

1.1. Importance of Requirements Engineering

A requirement, as defined by the International Council of Systems Engineering (INCOSE), is "a statement that identifies a system, product or process characteristic or

constraint, which is unambiguous, clear, unique, consistent, standalone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability" [1]. Requirements represent the first step towards designing systems, products, and enterprises. As such, the requirements should be [2,3]:

- **Necessary:** capable of conveying what is necessary to achieve the required system functionalities while being compliant with regulations;
- **Clear:** able to convey the desired goal to the stakeholders by being simple and concise;
- **Traceable:** able to be traced back to higher-level specifications and vice versa;
- **Verifiable:** can be verified by making use of different verification processes, such as analysis, inspection, demonstration, and test;
- **Complete:** the requirements should result in a system that successfully achieves the client's needs while being compliant with the regulatory standards.

The current workflow of requirements engineering is typically to elicit customer input via a series of design studies and interviews. These serve to understand the top needs of customers and what they value in a potential system. These needs are mapped into a technical language of functions, performance attributes, and "-ilities" (availability, maintainability, and reliability) that can be inspected by engineers to initiate more detailed design efforts. Sometimes formal mapping methods such as Quality Function Deployment (QFD) are used, but more often, this is a crafts approach developed within system-developing organizations.

Requirements can be of different types, such as functional, non-functional, design, performance, certification, etc., based on the system of interest [4]. This classification has been developed to help focus the efforts of requirements developers and help them provide guidelines on the style and structure of different requirements. Requirements engineering has developed as a domain that involves elucidating stakeholder expectations and converting them into technical requirements [5]. In other words, it involves defining, documenting, and maintaining requirements throughout the engineering lifecycle [6].

Well-defined requirements and good requirements engineering practices are critical to the successful design, development, and operation of systems, products, and processes. Errors during the requirement definition phase, on the other hand, can trickle down to downstream tasks such as system architecting, system design, implementation, inspection, and testing [7], leading to dramatic engineering and programmatic consequences when caught late in the product life cycle [8,9]. When "requirements engineering" is practiced as a separate effort on large teams, typically with a dedicated team or at least a lead, it becomes very process-focused. Configuration management, customer interviews, validation and verification planning, and maturation sessions all take place within the effort. Specialized software packages such as DOORS have been crafted over many years to service the needs of processing requirements as a collection of individual data records. However, one connection that is often lost is that between the requirements development team and the architectural team. Because Natural Language (NL) is predominantly used to write requirements [10], requirements are commonly prone to ambiguities and inconsistencies. This, in turn, increases the likelihood of errors and issues in the way requirements are formulated. The ambiguities of the requirements text are eventually resolved, and not entirely satisfactorily, in the test definition. At this point, misunderstandings and misses are quite expensive. If the misses do not prevent the system from being fielded, they will often be overlooked and instead simply become potentially lost value to the end user. This overall process orientation can make requirements development look like it is just part of the paperwork overhead of delivering large projects to institutional customers rather than the vital part of the understanding and embodiment of customer needs.

1.2. Shift towards Model-Based Systems Engineering

Most of the systems in the present-day world are complex and hence need a comprehensive approach to their design and development [11]. To accommodate this need, there has been a drive toward the development and use of Model-Based Systems Engineering

(MBSE) principles and tools, where activities that support the system design process are accomplished using models compared to traditional document-based methods [12]. Models capture the requirements as well as the domain knowledge and make them accessible to all stakeholders [13,14]. While MBSE shows great promise, the ambiguities and inconsistencies inherent to NL requirements hinder their direct conversion to models [15]. Hand-crafting models is time-consuming and requires highly specialized subject matter expertise. As a result, there is a need to convert NL requirements into a semi-machine-readable form so as to facilitate their integration and use in an MBSE environment. The need to access data within requirements rather than treating the statement as a standalone object has also been recognized by the International Council on System Engineering's (INCOSE) Requirements Working Group in their recent publication [16]. The document envisions an informational environment where requirements are not linked only to each other or test activities but also to architectural elements. This is represented in the figure below, which envisions a web of interconnections from a new kind of requirement object, which is a fusion of the natural language statement and machine-readable attributes that can connect to architectural entities such as interfaces and functions.

The approach of creating and maintaining requirements as more information-rich objects than natural language sentences has been called "Information-Based Needs and Requirements Definition and Management" (I-NRDM). In the INCOSE manual [16], it is recommended that model-based design (working on architecture and analysis) be combined with I-NRDM to be the full definition of MBSE. The "Property-Based Requirement" of recent SysML revisions can serve as the "Requirements Expression", as shown in Figure 1.

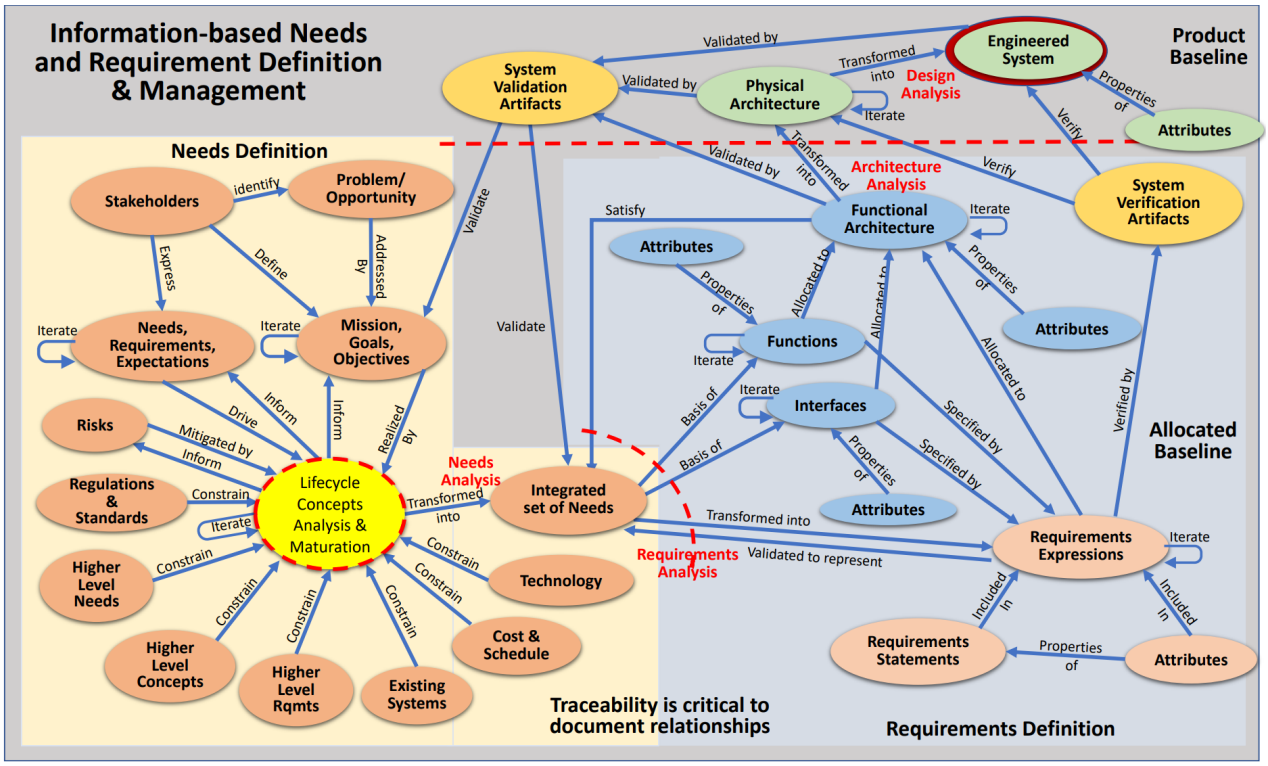

Figure 1. Information-based requirement development and management model [17].

Despite these identified needs and ongoing developments, there are no standard tools or methods for converting NL requirements into a machine-readable/semi-machine-readable form.

1.3. Research Focus and Expected Contributions

A first step towards standardizing requirements consists in classifying them according to their type (design, functional, performance, etc.). For example, if a certain requirement is

classified as a *design requirement*, then it is expected to contain certain specific information regarding the system design as compared to a *functional requirement*, for example, which focuses on the functions to be achieved by a system. Hence, it is important to be able to tell one requirement apart from another. While the field of Natural Language Processing (NLP) has shown promise when it comes to classifying text, text classification has mainly been limited to sentiment analysis, news genre analysis, and movie review analysis, and when applied to engineering disciplines, has mostly focused on software requirements analysis. In other words, its application to aerospace engineering requirements and, in particular, requirements centered around certification is limited, if not entirely lacking. To address the aforementioned need, the research presented herein develops an annotated aerospace requirements corpus, leverages a pre-trained language model, the Bidirectional Encoder Representational Transformer (BERT) [18], and fine-tunes it on the aforementioned labeled dataset of aerospace requirements for the purpose of requirement classification. This model is then further benchmarked against existing models. As a result, the main contributions of this work are as follows:

1. Creation of the first open-source annotated aerospace requirements dataset. This dataset includes three types of requirements, namely, design, functional, and performance.
2. Demonstration of a methodology describing the data collection, cleaning, and annotation of aerospace requirements from Parts 23 and 25 of Title 14 Code of Federal Regulations (CFRs) [19]. This demonstration is particularly important as it is missing from the existing literature on the use of NLP on datasets.
3. Demonstration of the fine-tuning of a pre-trained large language model (BERT) to obtain an aerospace requirements classifier (aeroBERT-Classifier), which generalizes despite having been trained on a small annotated dataset.
4. Demonstration of the viability of LMs in classifying high-level policy requirements such as the Federal Airworthiness Requirements (FARs), which have resisted earlier attempts to use NLP for automated processing. These requirements are particularly freeform and often complex expressions with a higher degree of classification difficulty than typical software or systems requirements (even for expert practitioners).

The remainder of this paper is organized as follows: Section 2 presents the relevance of NLP in requirements engineering and discusses various language models for text classification. Section 3 identifies the gaps in the literature and summarizes the research goal. Section 4 presents the technical approach taken to collect data, create a requirements corpus, and fine-tune different variants of BERT. Section 5 discusses the results and compares the results from the aeroBERT-Classifier with that of GPT-2, Bi-LSTM (using GloVe [20]), and bart-large-mnli (zero-shot text classification). Lastly, Section 6 summarizes the research conducted as part of this effort and discusses potential avenues for future work.

2. Background

This section provides the necessary background into the historical and current-day use of NLP in the field of requirements engineering. The following sections and subsections describe the evolution of NLP and establish the relevance of pre-trained large language models (LLMs) to requirements classification.

2.1. Natural Language Processing for Requirements Engineering (NLP4RE)

Requirements are almost always written in NL [21] to make them accessible to different stakeholders. According to various surveys, NL was deemed to be the best way to express requirements [22], and 95% of 151 software companies surveyed revealed that they were using some form of NL to capture requirements [23]. Given the ease of using NL for requirements elicitation, researchers have been striving to come up with NLP tools for requirements processing dating back to the 1970s. Tools such as the Structured Analysis Design Technique (SADT) and the System Design Methodology (SDM) developed at MIT are systems that were created to aid in requirement writing and management [22]. Despite

the interest in applying NLP techniques and models to the requirements engineering domain, the slow development of natural language technologies thwarted progress until recently [10]. The availability of NL libraries/toolkits (Stanford CoreNLP [24], NLTK [25], spaCy [26], etc.) and off-the-shelf transformer-based [27] pre-trained language models (LMs) (BERT [18], BART [28], etc.) have propelled NLP4RE into an active area of research.

A recent survey performed by Zhao et al. reviewed 404 NLP4RE studies conducted between 1983 and April 2019 and reported on the developments in this domain [29]. Among those 404 NLP4RE studies, 370 were analyzed and classified based on the main NLP4RE task being performed. It was found that 26.22% of these studies focused on detecting linguistic errors in requirements (use of ambiguous phrases, conformance to boilerplates, etc.), 19.73% focused on requirements classification, and 17.03% of text extraction tasks focused on the identification of key domain concepts. Figure 2 shows a clear increase in the number of published studies in NLP4RE over the years. This underlines the critical role that NLP plays in requirements engineering, a role that is expected to become more important with time as the availability of off-the-shelf language models increases.

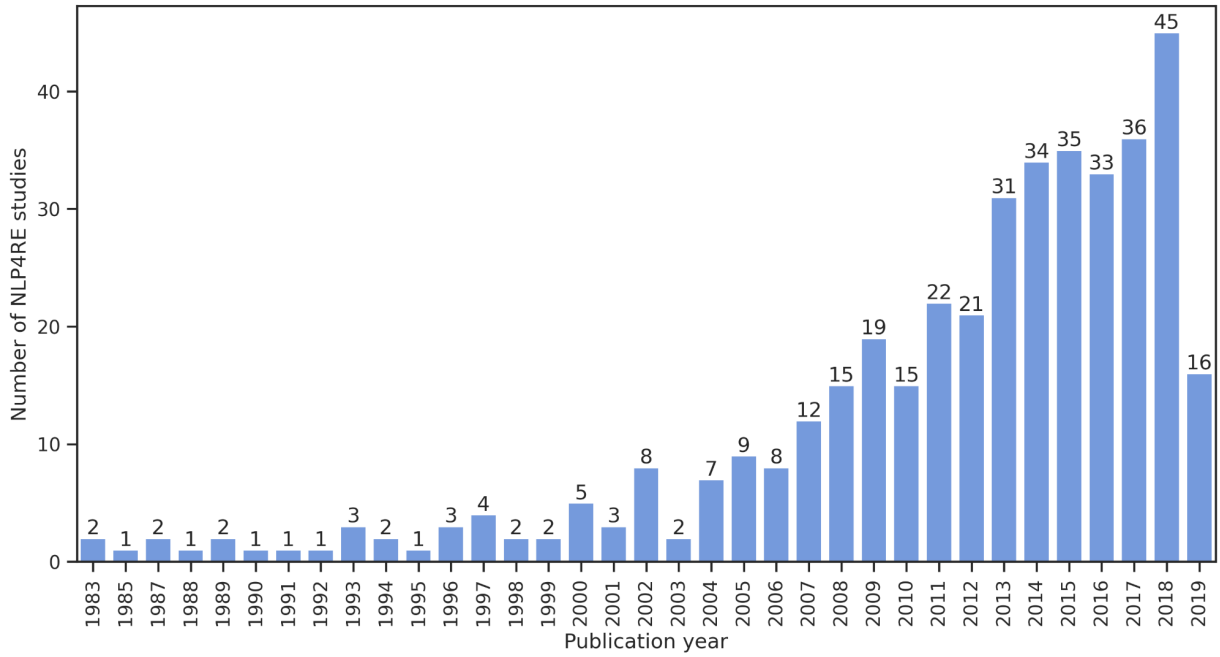

Figure 2. Published studies in the NLP4RE domain between 1983 and April 2019 [29].

The following section focuses more specifically on requirement classification, which is a critical step toward their conversion into a semi-machine-readable/standardized format. In particular, it discusses how NLP has evolved and can be leveraged to enable classification.

2.2. Natural Language Processing (NLP) and Language Models (LMs)

NLP is promising when it comes to classifying requirements, which is an anticipated first step in the development of pipelines that can convert free-form NL requirements into standardized requirements in a semi-automated manner. Language models (LMs), in particular, can be leveraged for classifying text (or requirements, in the context of this research) [18,28,30]. Language modeling was classically defined as the task of predicting which word comes next [31]. Initially, this was limited to statistical language models [32], which use prior word sequences to compute the conditional probability for each of a future vocabulary word. The high-dimensional discrete language representations limit these models to N-grams [33], where only the prior N words are considered for predicting the next word or short sequences of following words, typically using high-dimensional one-hot encodings for the words.

Neural LMs came into existence in the 2000s [34] and leveraged neural networks to simultaneously learn lower-dimensional word embeddings and learn to estimate conditional probabilities of the next words simultaneously using gradient-based supervised learning. This opened the door to ever-more-complex and effective language models to perform an expanding array of NLP tasks, starting with distinct word embeddings [35] to recurrent neural networks (RNNs) [36] and LSTM encoder–decoders [37] to attention mechanisms [38]. These models did not stray too far from the N-gram statistical language modeling paradigm, with advances that allowed text generation beyond a single next word with, for example, the beam search in [37] and sequence-to-sequence learning in [39]. These ideas were applied to distinct NLP tasks.

In 2017, the Transformer [27] architecture was introduced that improved computational parallelization capabilities over recurrent models and therefore enabled the successful optimization of larger models. Transformers consist of stacks of encoders (encoder block) and stacks of decoders (decoder block), where the encoder block receives the input from the user and outputs a matrix representation of the input text. The decoder takes the input representation produced by the encoder stack and generates outputs iteratively [27].

All of these works required training on a single labeled dataset for a specific task. In 2018–2020, several new models emerged and set new state-of-the-art marks in nearly all NLP tasks. These transformer-based models include the Bidirectional Encoder Representational Transformer (BERT) language model [18] (auto-encoding model), the Generative Pre-trained Transformer (GPT) family [40,41] of auto-regressive language models, and T5 character-level language model [42]. These sophisticated language models break the single dataset-single task modeling paradigm of most mainstream models in the past. They employ self-supervised pre-training on massive *unlabeled* text corpora. For example, BERT is trained on Book Corpus (800 M words) and English Wikipedia (2500 M words) [18]. Similarly, GPT-3 is trained on 500 B words gathered from datasets of books and the internet [41]. These techniques set up automated supervised learning tasks, such as masked language modeling (MLM), next-sentence prediction (NSP), and generative pre-training. No labeling is required as the labels are automatically extracted from the text and hidden from the model, and the model is trained to predict them. This enables the models to develop a deep understanding of language, independent of the NLP task. These pre-trained models are then fine-tuned on much smaller labeled datasets, leading to advances in the state of the art (SOTA) for nearly all downstream NLP tasks, such as Named-Entity Recognition (NER), text classification, language translation, and question answering, etc. [18,43–45].

Training these models from scratch can be prohibitive given the computational power required; to put things into perspective, it took 355 GPU-years and USD 4.6 million to train GPT-3 [46]. Similarly, the pre-training cost for THE $\text{BERT}_{\text{BASE}}$ model with 110 million parameters varied between USD 2.5k and USD 50k, while it cost between USD 10k and USD 200k to pre-train $\text{BERT}_{\text{LARGE}}$ with 340 million parameters [47]. Pre-training the BART LM took 12 days with 256 GPUs [28]. These general-purpose LMs can then be fine-tuned for specific downstream NLP tasks at a small fraction of the computational cost and time—even in a matter of minutes or hours on a single-GPU computer [18,48–51].

Pre-trained LMs are readily available on the Hugging Face platform [52] and can be accessed via the *transformers* library. These models can then be fine-tuned on downstream tasks with a labeled dataset specific to the downstream task of interest. For example, for text classification, a dataset containing the text and the corresponding labels should be used for fine-tuning the model. The final model with text classification capability will eventually have different model parameters as compared to the parameters it was initialized with [18]. Figure 3 illustrates the pre-training and fine-tuning steps for BERT for a text classification task.

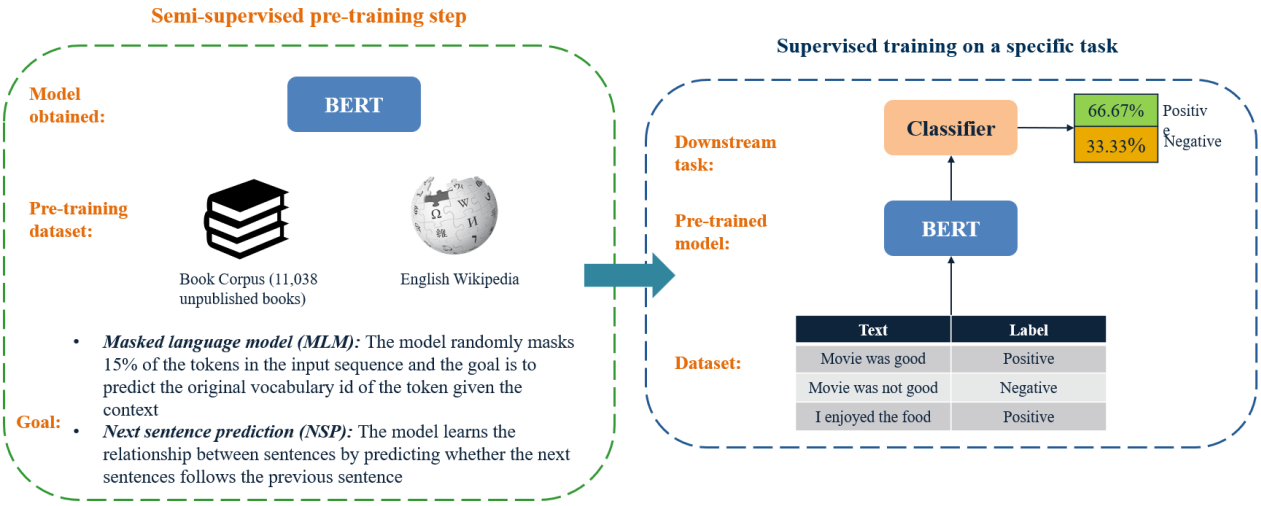

Figure 3. Fine-tuning a BERT language model on a text classification task [18,53].

Multiple approaches to text classification exist. The following section discusses the use of Bidirectional Encoder Representations from Transformers (BERT) and Zero-shot Text Classification for this purpose.

2.2.1. Bidirectional Encoder Representations from Transformers (BERT)

As mentioned previously, BERT is a pre-trained LM that is capable of learning deep *bidirectional representations* from an unlabeled text by jointly incorporating both the left and right context of a sentence in all layers [18]. Being a transformer-based LM, BERT is capable of learning the complex structure and the non-sequential content in language by using *attention mechanisms*, fully-connected neural network layers, and positional encoding [27,53]. Despite being trained on a large corpus, the vocabulary of BERT is just 30,000 words since it uses WordPiece Tokenizer, which breaks words into sub-words and eventually into letters (if required) to accommodate for out-of-vocabulary words. This makes the practical vocabulary BERT can understand much larger. For example, "ing" is a single word-piece, so it can be added to nearly all verbs in the base vocabulary to extend the vocabulary tremendously. BERT comes in two variants when it comes to model architecture [18]:

1. **BERT$_{\mathbf{BASE}}$**: contains 12 encoder blocks with a hidden size of 768 and 12 self-attention heads (total of 110 M parameters);
2. **BERT$_{\mathbf{LARGE}}$**: contains 24 encoder blocks with a hidden size of 1024 and 16 self-attention heads (total of 340 M parameters).

Both BERT variants come in *cased* and *uncased* versions. In the case of the *cased* model, the input text stays the same (containing both upper and lowercase letters). However, the text is made lowercase before being tokenized by a WordPiece tokenizer in the *uncased* model.

BERT is pre-trained on two self-supervised language tasks, namely, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) (Figure 3), which help it develop a general-purpose understanding of natural language. However, it can be fine-tuned to perform various downstream tasks such as text classification, NER, Part-of-Speech (POS) tagging, etc. A task-specific output layer is what separates a pre-trained BERT from a BERT fine-tuned to perform a specific task.

In a recent study, Hey et al. fine-tuned the BERT language model on the PROMISE NFR dataset [54] to obtain NoRBERT (Non-functional and functional Requirements classification using BERT)—a model capable of classifying requirements [45]. NoRBERT is capable of performing four tasks, namely, (1) binary classification of requirements into two classes (functional and non-functional); (2) binary and multi-class classification of four non-functional requirement classes (usability, security, operational, and performance); (3) multi-class classification of ten non-functional requirement types; and (4) binary clas-

sification of the functional and quality aspects of requirements. NoRBERT was able to achieve an average F1 score of 0.87 on the most frequently occurring classes of requirements (usability, performance, operational, and security). In particular, it demonstrates the relevance and potential of transfer-learning approaches to requirements engineering research as a means to address the limited availability of labeled data. The PROMISE NFR dataset [54], which was used in [45], contains 625 requirements in total (255 functional and 370 non-functional, which are further broken down into different "sub-types"). Table 1 provides some examples from the PROMISE NFR dataset.

Table 1. Requirements examples from the PROMISE NFR dataset [54].

Serial No.	Requirements
1	The product shall be available for use 24 h per day 365 days per year.
2	The product shall synchronize with the office system every hour.
3	The system shall let existing customers log into the website with their email address and password in under 5 s.
4	The product should be able to be used by 90% of novice users on the internet.
5	The ratings shall be on a scale of 1–10.

These requirements originated from 15 projects written by students and, as a result, might not have been written according to industry standards. The PROMISE NFR dataset is, to the authors' knowledge, the only requirements dataset of its kind that is publicly available. However, this dataset was not deemed suitable for this work because it predominantly focuses on software engineering systems and requirements.

Another avenue for text classification is zero-shot text classification, which is discussed in detail in the next section.

2.2.2. Zero-Shot Text Classification

Models performing a task they have not been explicitly trained for is called zero-shot learning (ZSL) [55]. Such models, due to their nature, provide a logical basis for the evaluation of LMs that have been previously fine-tuned on requirements specific to a discipline of interest.

There are two general ways for ZSL, namely, *entailment-based* and *embedding-based* methods. Yin et al. proposed a method for zero-shot text classification using pre-trained Natural Language Inference (NLI) models [56]. The *bart-large-mnli* model was obtained by training *bart-large* [28] on the MultiNLI (MNLI) dataset, which is a crowd-sourced dataset containing 433,000 sentence pairs annotated with textual entailment information (0: entailment; 1: neutral; 2: contradiction) [57]. For example, to classify the sentence "*The United States is in North America*" into one of the possible classes, namely, *politics*, *geography*, or *film*, we could construct a hypothesis such as *This text is about geography*. The probabilities for the entailment and contraction of the hypothesis will then be converted into probabilities associated with each of the labels provided.

Alhosan et al. [58] performed a preliminary study for the classification of requirements using ZSL in which they classified non-functional requirements into two categories, namely usability and security. An embedding-based method was used where the probability of the relatedness (cosine similarity) between the text embedding layer and the tag (or label) embedding layer was calculated to classify the requirement into either of the two categories. This work, which leveraged a subset of the PROMISE NFR dataset, was able to achieve a recall and F-score of 82%. The authors acknowledge that certain LMs ($\text{RoBERTa}_{\text{Base}}$ and XLM-RoBERTa) seemed to favor one or the other class, hence classifying all the requirements as either usability or security. This was attributed to the fact that LLMs are trained on general domain data and hence might not perform well on specialized domains [54].

3. Research Gaps and Objectives

As stated previously, SOTA LLMs are typically pre-trained on general-domain text corpora, such as news articles, Wikipedia, book corpora, movie reviews, Tweets, etc. Off-the-shelf models for specific tasks such as NER or text classification are fine-tuned on generic text datasets as well. While the underlying language understanding learned during pre-training is still valuable, the fine-tuned models are less effective when it comes to working with specialized language representative of technical domains, such as aerospace, healthcare, etc. For example, using a language model fine-tuned on classifying sentiment data will not be effective if used for classifying aerospace requirements. To illustrate this, bart-large-mnli LM was used for classifying the aerospace requirement "*The installed powerplant must operate without any hazardous characteristics during normal and emergency operation within the range of operating limitations for the airplane and the engine*" into three classes (design requirement, functional requirement, and performance requirement), as shown in Figure 4. The true label associated with the requirement is performance; however, the zero-shot text classification model classified it as a functional requirement. Several other examples were evaluated, and a skew toward the functional requirement class was observed.

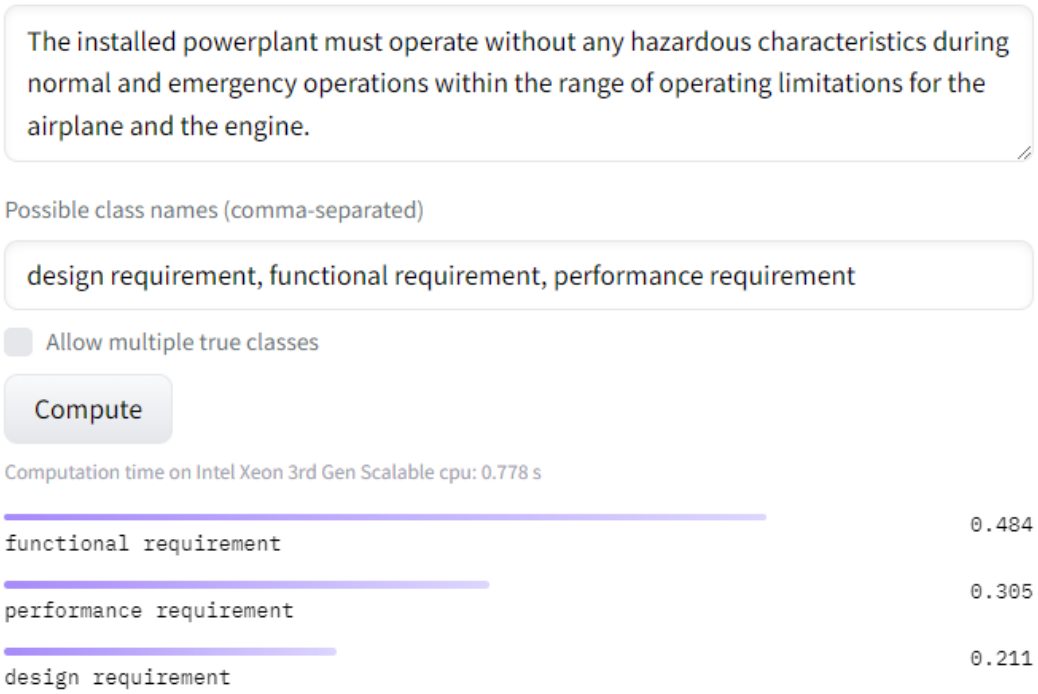

Figure 4. Classification of a performance requirement by the bart-large-mnli model [56].

The existence of models such as *aeroBERT-NER* [59], *SciBERT* [60], *FinBERT* (Financial BERT) [61], *BioBERT* [62], *ClinicalBERT* [63], and *PatentBERT* [64] stresses the importance of domain-specific corpora for pre-training and fine-tuning LMs when it comes to domains that are not well represented in the general-domain text.

Open-source datasets, however, are scarce when it comes to requirements engineering, especially those specific to aerospace applications. This has hindered the use of modern-day language models for requirements classification tasks. In addition, because annotating requirements datasets requires subject-matter expertise, crowd-sourcing as a means of data collection is not a viable approach.

Summary

The overarching goal of this research is eventually the development of a methodology for the standardization of aerospace requirements into semi-machine-readable requirements by making use of requirements boilerplate. Figure 5 shows the pipeline consisting of different language models, the outputs of which, put together, are expected to facilitate the conversion of NL aerospace requirements into semi-machine-readable requirements, which will inform boilerplate creation.

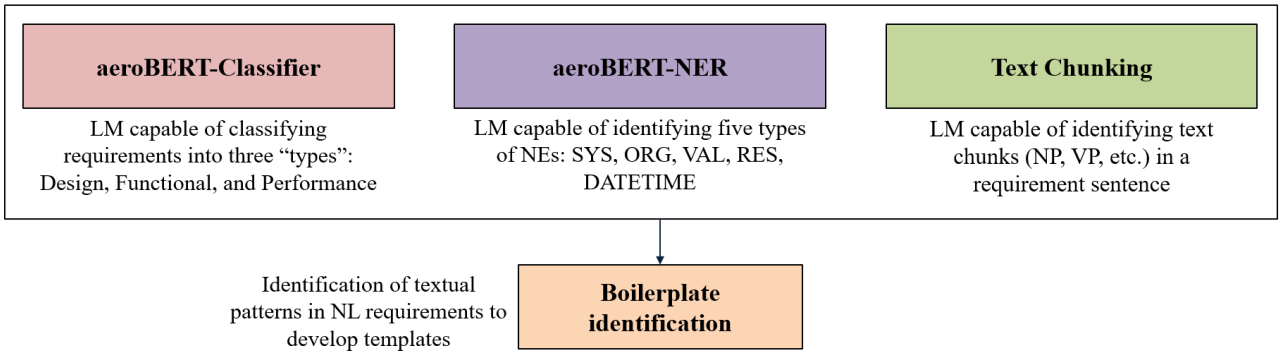

Figure 5. Pipeline for converting NL requirements to semi-machine-readable form enabling boilerplate creation.

A critical first step in this pipeline is the appropriate classification of aerospace requirements, which is the focus of this paper. To that end, the present work concentrates on fine-tuning a language model for classifying aerospace requirements into various types. This is achieved through the:

1. **Creation of a labeled aerospace requirements corpus:** Aerospace requirements are collected from Parts 23 and 25 of Title 14 CFRs [65] and annotated. The annotation involves labeling each requirement with its type (e.g., functional, performance, interface, design, etc.).
2. **Fine-tuning of BERT for aerospace requirements classification:** The annotated aerospace requirements are used to fine-tune several variants of the BERT LM ($BERT_{LARGE-UNCASED}$, $BERT_{LARGE-CASED}$, $BERT_{BASE-UNCASED}$, and $BERT_{BASE-CASED}$). We call the best resulting model aeroBERT-Classifier. Metrics such as precision, recall, and F1 score are used to assess the model performance.
3. **The comparison of the performance of aeroBERT-Classifier against other text classification models:** The performance of aeroBERT-Classifier is compared to that of GPT-2 and Bi-LSTM (with GloVe word embedding), which are also trained or fine-tuned on the developed aerospace requirements dataset. Lastly, the model performance is compared to results obtained by using *bart-large-mnli*, a zero-short learning classifier, to further emphasize the importance of transfer learning in low-resource domains such as aerospace requirements engineering.

The following section describes the methodology for this work.

4. Materials and Methods

Aerospace requirements are often always proprietary, meaning that they are of limited open-source availability. Since large amounts of data are required to train an LM from scratch, this research focuses instead on *fine-tuning* the BERT variants for the classification of aerospace requirements. To that end, an aerospace corpus containing labeled requirements is developed and used.

The methodology for developing the aeroBERT-Classifier is illustrated in Figure 6. The following sections describe each step in detail.

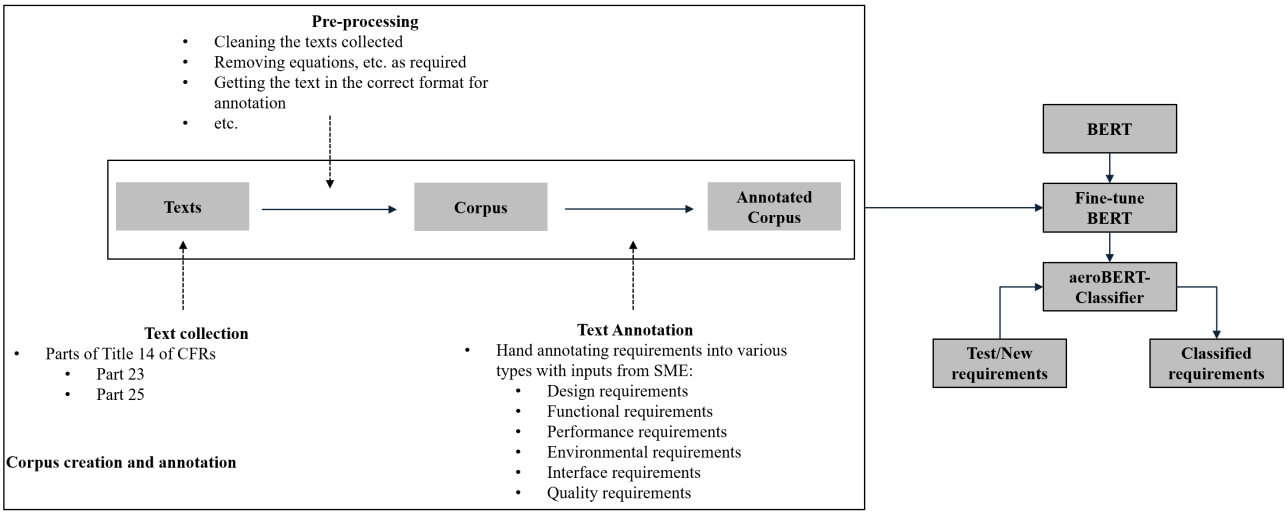

Figure 6. Methodology for obtaining aeroBERT-Classifier.

4.1. Data Collection, Cleaning, and Annotation

The creation of an annotated aerospace requirements corpus is a critical step since such a corpus is not readily available in the open-source domain and is required for fine-tuning BERT. As seen in Figure 6, the first step to creating a requirements dataset is to collect aerospace requirements from various sources. Table 2 provides a list of the resources leveraged to obtain requirements for the purpose of creating a classification corpus.

Table 2. Resources used for the creation of aerospace requirements classification corpus.

Serial No.	Name of Resource (Title 14 CFR)
1	Part 23: Airworthiness Standards: Normal, Utility, Acrobatic and Commuter Airplanes
2	Part 25: Airworthiness Standards: Transport Category Airplanes

Requirements oftentimes appear in paragraph form, which requires some rewriting to convert texts/paragraphs into requirements that can then be added to the dataset. For example, Table 3 shows 14 CFR §23.2145(a) in its original and modified form, which resulted in three distinct requirements. A total of 325 requirements were collected and added to the classification corpus.

Table 3. Example showing the modification of text from 14 CFR §23.2145(a) into requirements.

14 CFR §23.2145(a)	Requirements Created
Airplanes not certified for aerobatics must:	
(1) Have static longitudinal, lateral, and directional stability in normal operations;	**Requirement 1:** Airplanes not certified for aerobatics must have static longitudinal, lateral, and directional stability in normal operations.
(2) Have short dynamic period and Dutch roll stability in normal operations; and	**Requirement 2:** Airplanes not certified for aerobatics must have short dynamic period and dutch roll stability in normal operations.
(3) Provide stable control force feedback throughout the operating envelope.	**Requirement 3:** Airplanes not certified for aerobatics must provide stable control force feedback throughout the operating envelope.

After obtaining the requirements corpus, a few other changes were made to the text, as shown in Table 4. The symbols '§' and '§§' were replaced with the word 'Section' and 'Sections', respectively, to make it more intuitive for the LM to learn patterns. Similarly, the dots '.' occurring in the section names were replaced with '-' to avoid confusing them with sentence endings. The above pre-processing steps were also used in a companion article for obtaining annotated named entities (NEs) corpus to train aeroBERT-NER [59].

Table 4. Symbols that were modified for corpus creation [59].

Original Symbol	Modified Text/Symbol	Example
§	Section	§25.531 → Section 25.531
§§	Sections	§§25.619 through 25.625 → Sections 25.619 through 25.625
Dot ('.') used in section numbers	Dash ('-')	Section 25.531 → Section 25–531

After pre-processing the requirements text, the next step consisted of annotating the requirements based on their type. Requirements were classified into six categories, namely, Design, Functional, Performance, Interface, Environmental, and Quality. The definitions used for the requirement types, along with some examples, are provided in Table 5. Our author base consists of the expertise required to make sure that the requirements were annotated correctly into various categories. It is important to keep in mind that different companies/industries might have their own definitions for requirements specific to their domain.

Table 5. Definitions used for labeling/annotating requirements [2,5,66].

Requirement Type	Definition
Design	Dictates "how" a system should be designed given certain technical standards and specifications; **Example:** Trim control systems must be designed to prevent creeping in flight.
Functional	Defines the functions that need to be performed by a system in order to accomplish the desired system functionality; **Example:** Each cockpit voice recorder shall record voice communications of flight crew members on the flight deck.
Performance	Defines "how well" a system needs to perform a certain function; **Example:** The airplane must be free from flutter, control reversal, and divergence for any configuration and condition of operation.
Interface	Defines the interaction between systems [67]; **Example:** Each flight recorder shall be supplied with airspeed data.
Environmental	Defines the environment in which the system must function; **Example:** The exhaust system, including exhaust heat exchangers for each powerplant or auxiliary power unit, must be designed to prevent likely hazards from heat, corrosion, or blockage.
Quality	Describes the quality, reliability, consistency, availability, usability, maintainability, and materials and ingredients of a system [68]; **Example:** Internal panes must be made of nonsplintering material.

As mentioned previously, the corpus includes a total of 325 aerospace requirements. 134 of these 325 requirements (41.2%) were annotated as Design requirements, 91 (28%) as Functional Requirements, and 62 (19.1%) as Performance requirements. Figure 7 provides a breakdown by requirement type.

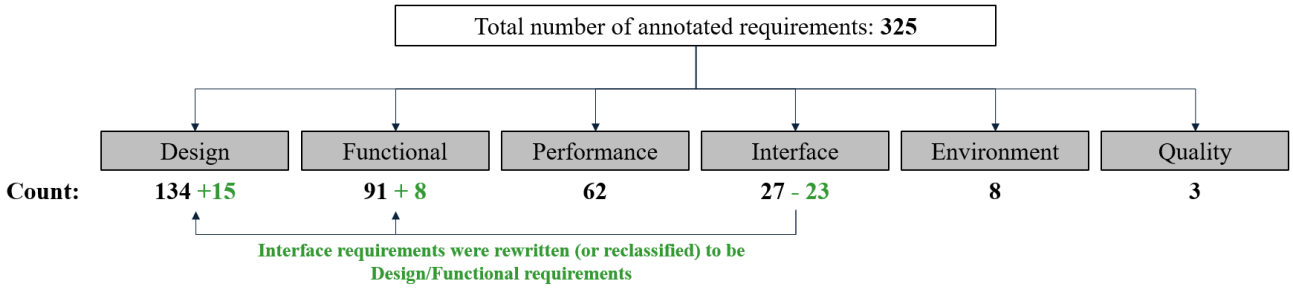

Figure 7. Six types of requirements were initially considered for the classification corpus. Due to the lack of sufficient examples for Interface, Environment, and Quality requirements, these classes were dropped at a later phase. However, some of the Interface requirements (23) were rewritten (or reclassified) to convert them into either Design or Functional requirements to include them in the final corpus, which only contains Design, Functional, and Performance requirements.

As seen in Figure 7, the dataset is skewed toward Design, Functional, and Performance requirements (in that order). Since the goal is to develop an LM that is capable of classifying requirements, a balanced dataset is desired, which is not the case here. As seen, there are not enough examples of Interface, Environment, and Quality requirements in the primary data source (Parts 23 and 25 of Title 14 of the Code of Federal Regulations (CFRs)). This can be due to the fact that Interface, Environment, and Quality requirements do not occur alongside the other types of requirements.

In order to obtain a more balanced dataset, Environment and Quality requirements were dropped completely. However, some of the Interface requirements (23) were rewritten (or reclassified) as Design and Functional requirements, as shown in Table 6. The rationale for this reclassification was that it is possible to treat the interface as a thing being specified rather than as a special requirement type between two systems.

Table 6. Examples showing the modification of Interface requirements into other types of requirements.

Original Interface Requirement	Modified Requirement Type/Category
Each flight recorder shall be supplied with airspeed data.	The airplane shall supply the flight recorder with airspeed data. (**Functional Requirement**)
Each flight recorder shall be supplied with directional data.	The airplane shall supply the flight recorder with directional data. (**Functional Requirement**)
The state estimates supplied to the flight recorder shall meet the aircraft-level system requirements and the functionality specified in Section 23–2500.	The state estimates supplied to the flight recorder shall meet the aircraft level system requirements and the functionality specified in Section 23–2500. (**Design Requirement**)

The final classification dataset includes 149 Design requirements, 99 Functional requirements, and 62 Performance requirements (see Figure 7). Lastly, the 'labels' attached to the requirements (design requirement, functional requirement, and performance requirement) were converted into numeric values: 0, 1, and 2, respectively. The final form of the dataset is shown in Table 7.

Table 7. Classification dataset format.

Requirements	Label
Each cockpit voice recorder shall record voice communications transmitted from or received in the airplane by radio.	1
Each recorder container must be either bright orange or bright yellow.	0
Single-engine airplanes, not certified for aerobatics, must not have a tendency to inadvertently depart controlled flight.	2
Each part of the airplane must have adequate provisions for ventilation and drainage.	0
Each baggage and cargo compartment must have a means to prevent the contents of the compartment from becoming a hazard by impacting occupants or shifting.	1

4.2. Preparing the Dataset for Fine-Tuning BERT

Language models (LMs) expect inputs in a certain format, which may vary from one LM to another. BERT expects inputs in the format discussed below [18]:

Special tokens:

- **[CLS]:** This token is added to the beginning of every sequence of text, and its final hidden state contains the aggregate sequence representation for the entire sequence, which is then used for the sequence classification task.
- **[SEP]:** This token is used to separate one sequence from the next and is needed for Next-Sentence-Prediction (NSP) task. Since aerospace requirements used for this research are single sentences, this token was not used.
- **[PAD]:** This token is used to make sure that all the input sequences are of the same length. The maximum length for the input sequences was set to 100 after examining the distribution of lengths of all sequences in the training set (Figure 8). All the sequences with a length less than the set maximum length will be post-padded with [PAD] tokens till the sequence length is equal to the maximum length. The sequences that are longer than 100 will be truncated.

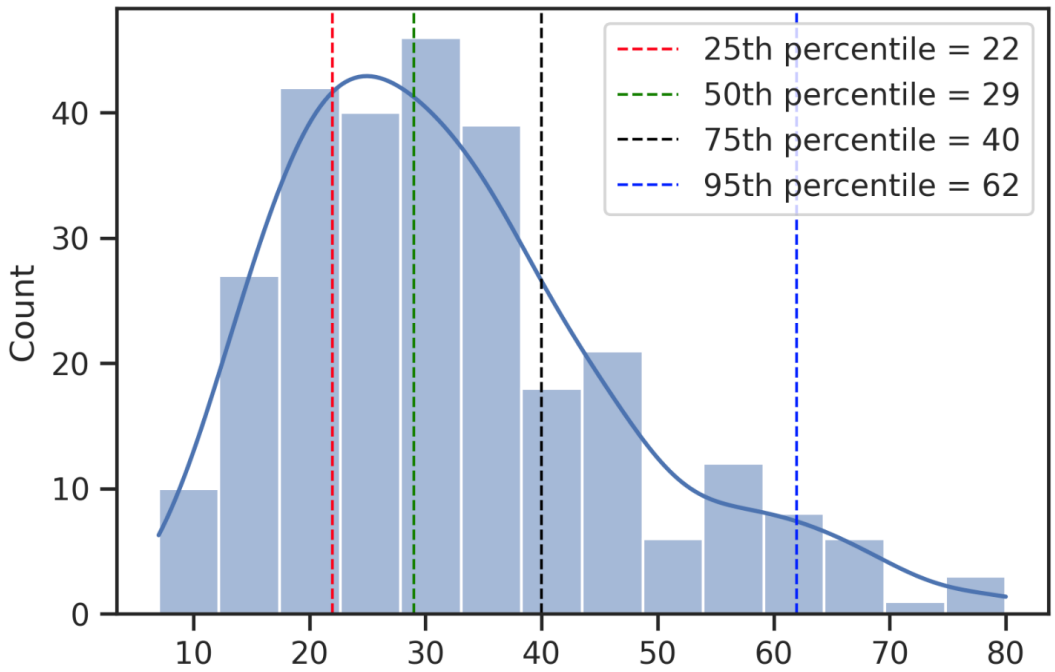

Figure 8. Figure showing the distribution of sequence lengths in the training set. The 95th percentile was found to be 62. The maximum sequence length was set to 100 for the aeroBERT-Classifier model.

All the sequences were *cased* or *uncased* depending on the BERT variant being fine-tuned. This was followed by a tokenization step where a WordPiece tokenizer was used to split requirement sentences into tokens. Special tokens ([CLS], and [PAD]) were added to the beginning and end of the requirements, and each token was mapped to its respective ID in the BERT vocabulary of approximately 30,000 tokens. Lastly, the requirements sentences were padded/truncated to match the maximum length specified, and attention masks were created (1 for "real" tokens, and 0 for [PAD] tokens).

The dataset was split into training (90%) and test set (10%) containing 279 and 31 samples, respectively (the corpus contains a total of 310 requirements). Table 8 gives a detailed breakdown of the count of each type of requirement in the training and test sets. The LM was fine-tuned on the training set, whereas the model performance was tested on the test set, which the model had not been exposed to during training.

Table 8. Breakdown of the types of requirements in the training and tests set.

Requirement Type	Training Set Count	Test Set Count
Design (0)	136	13
Functional (1)	89	10
Performance (2)	54	8
Total	**279**	**31**

4.3. Fine-Tuning BERT

A pre-trained BERT model with a linear classification layer on the top is loaded from the *Transformers* library from HuggingFace (`BertForSequenceClassification`). This model and the untrained linear classification layer (full-fine-tuning) are trained on the classification corpus created previously (Table 7).

The batch size was set to 16, and the model was trained for 20 epochs. The model was supplied with three tensors for training: (1) input IDs; (2) attention masks; and (3) labels for each example. The `AdamW` optimizer [69] with a learning rate of 2×10^{-5} was used. The previously calculated gradients were cleared before performing each backward pass. In addition, the norm of gradients was clipped to `1.0` to prevent the exploding gradient problem. The dropout rate was set to the default value of `0.1` (after experimenting with other rates) to promote the generalizability of the model and speed up the training process. The model was trained to minimize the cross-entropy loss function. In addition, the model performance on the test set was measured by calculating metrics, including the F1 score, precision, and recall. Various iterations of the model training and testing were carried out to make sure that the model was robust and reliable. The fine-tuning process took only 39 s on an NVIDIA Quadro RTX 8000 GPU with a 40 GB VRAM. The small training time can be attributed to the small training set. The optimizer, batch size, and other hyperparameters were tuned to maximize validation performance.

Figure 9 provides a rough schematic of the methodology used to fine-tune the BERT LM for the requirement, *"Trim control systems must be designed to prevent creeping in flight"*. The token embeddings are fed into the pre-trained LM, and the representations for these tokens are obtained after passing through 12 encoder layers. The representation for the first token ($R_{[CLS]}$) contains the aggregate sequence representation and is passed through a pooler layer (with a Tanh activation function) and then a linear classification layer. Class probabilities for the requirement belonging to the three categories (design, functional, and performance) are estimated, and the requirement is classified into the category with the highest estimated probability, *'Design'* in this case.

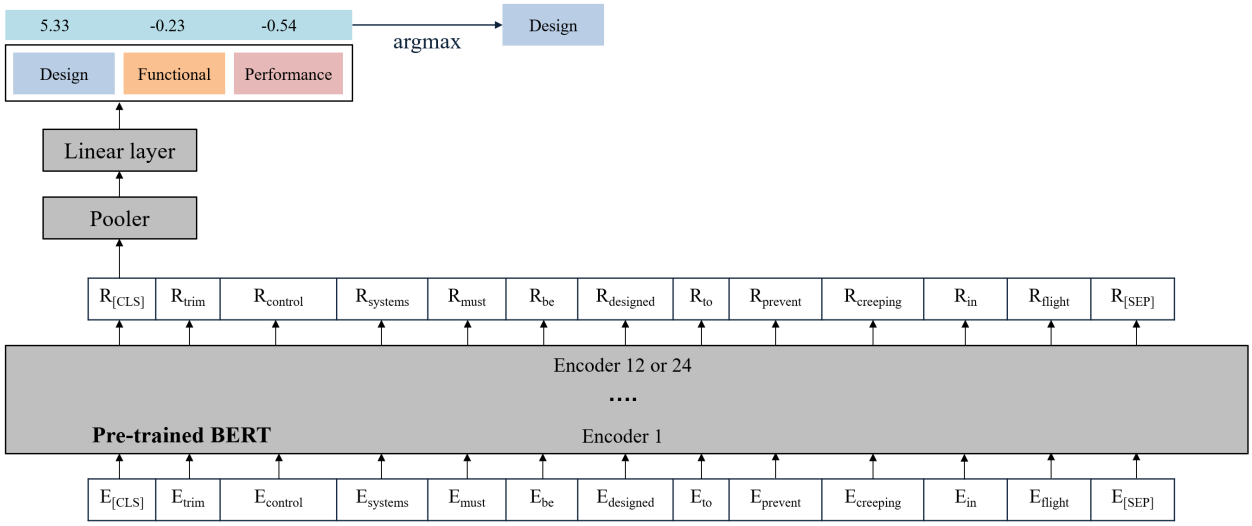

Figure 9. The detailed methodology used for the full-fine-tuning of BERT (base and large) LM is shown here. E_{name} represents the embedding for that particular WordPiece token, which is a combination of position, segment, and token embeddings. R_{name} is the representation for every token after it goes through the BERT model. Only $R_{[CLS]}$ is used for requirement classification since its hidden state contains the aggregate sequence representation [18].

5. Results

This section discusses the performance of aeroBERT-Classifier on the test dataset. In addition, the performance of this model is compared to other classification models, including fine-tuned GPT-2, trained Bi-LSTM (with GloVe), and zero-shot model bart-large-mnli.

5.1. aeroBERT-Classifier Performance

aeroBERT-Classifier was trained on a dataset containing 279 requirements. The dataset was imbalanced, meaning there was more of one type of requirement compared to others (136 design requirements compared to 89 functional, and 54 performance requirements). Accuracy as a metric will favor the majority class—design requirements in our case. Therefore, precision, recall, and F1 score were used to measure the model performance more rigorously. Table 9 provides the aggregate values for these metrics along with the breakdown for each requirement type. The aeroBERT-Classifier$_{LU}$ (obtained by fine-tuning BERT$_{LARGE\text{-}UNCASED}$) was able to identify 100% (Recall) of functional requirements present in the test set; however, of all the requirements that the model identified as functional requirements, only 83% (Precision) belonged to this category. Similarly, aeroBERT-Classifier$_{BC}$ (obtained by fine-tuning BERT$_{BASE\text{-}CASED}$) was able to identify 80% of all the functional requirements and 75% of all the performance requirements present in the test set. The precision obtained for functional and performance requirements was 0.89 and 0.86, respectively.

Table 9. Requirements classification results on aerospace requirements dataset. The highest score for each metric per class is shown in **bold**. (P = Precision, R = Recall, F1 = F1 Score).

Models	Design			Functional			Performance			Avg.
	P	R	F1	P	R	F1	P	R	F1	F1
aeroBERT-Classifier$_{LU}$	**0.91**	0.77	0.83	0.83	**1.0**	**0.91**	0.75	0.75	0.75	**0.83**
aeroBERT-Classifier$_{LC}$	0.86	**0.92**	**0.89**	0.82	0.90	0.86	0.83	0.63	0.71	0.82
aeroBERT-Classifier$_{BU}$	0.80	**0.92**	0.86	**0.89**	0.80	0.84	**0.86**	0.75	**0.80**	**0.83**
aeroBERT-Classifier$_{BC}$	0.79	0.85	0.81	0.80	0.80	0.80	**0.86**	0.75	**0.80**	0.80
GPT-2	0.67	0.60	0.63	0.67	0.67	0.67	0.70	**0.78**	0.74	0.68
Bi-LSTM (GloVe)	0.75	0.75	0.75	0.75	0.60	0.67	0.43	0.75	0.55	0.68
bart-large-mnli	0.43	0.25	0.32	0.38	0.53	0.44	0.0	0.0	0.0	0.34

Of all the variants of aeroBERT-Classifier evaluated, *uncased* variants outperformed the *cased* variants. This indicates that "casing" is not as important to text classification as it would be in the case of an NER task. In addition, the overall average F1 scores obtained by aeroBERT-Classifier$_{LU}$ and aeroBERT-Classifier$_{BU}$ were the same. This suggests that the *base-uncased* model is capable of learning the desired patterns in aerospace requirements in less training time and, hence, is preferred.

Various iterations of the model training and testing were performed, and the model performance scores were consistent. In addition, the aggregate precision and recall were not very far off from each other, giving rise to a high F1 score (harmonic mean of precision and recall). Since the difference between the training and test performance is low despite the small size of the dataset, it is expected that the model will generalize well to unseen requirements belonging to the three categories.

Table 10 provides a list of requirements from the test set that were misclassified (Predicted label $\neq$ Actual label) by aeroBERT-Classifier$_{BU}$. A confusion matrix summarizing the classification task is shown in Figure 10. It is important to note that some of the requirements were difficult to classify even by the authors with expertise in requirements engineering.

Table 10. List of requirements (from test set) that were misclassified by aeroBERT-Classifier$_{BU}$ (0: Design; 1: Functional; 2: Performance).

Requirements	Actual	Predicted
The installed powerplant must operate without any hazardous characteristics during normal and emergency operation within the range of operating limitations for the airplane and the engine.	2	1
Each flight recorder must be installed so that it remains powered for as long as possible without jeopardizing the emergency operation of the airplane.	0	2
The microphone must be located and, if necessary, the preamplifiers and filters of the recorder must be adjusted or supplemented so that the intelligibility of the recorded communications is as high as practicable when recorded under flight cockpit noise conditions and played back.	2	0
A means to extinguish a fire within a fire zone, except a combustion heater fire zone, must be provided for any fire zone embedded within the fuselage, which must also include a redundant means to extinguish a fire.	1	0
Thermal/acoustic materials in the fuselage must not be a flame propagation hazard.	1	0

The test set contained 13 design, 10 functional, and 8 performance requirements (Table 8). As seen in Table 10 and Figure 10, out of the 13 design requirements, only one was misclassified as a performance requirement. Of the eight performance requirements, two were misclassified. Two of the ten functional requirements were misclassified.

The training and testing were carried out multiple times, and the requirements shown in Table 10 were consistently misclassified, which might have been due to ambiguity in the labeling. Hence, it is important to have a *human-in-the-loop* (preferably a Subject Matter Expert (SME)) who can make a judgment call on whether a certain requirement was labeled wrongly or to support a requirement rewrite to resolve ambiguities.

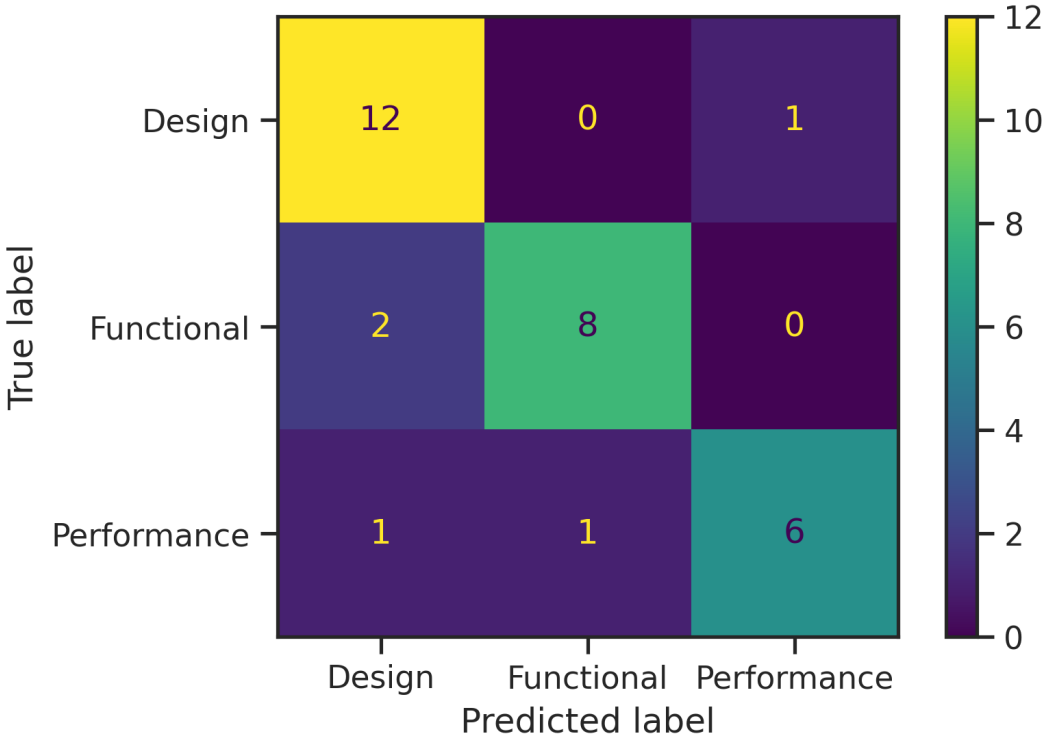

Figure 10. Confusion matrix showing the breakdown of the true and predicted labels by aeroBERT-Classifier$_{BU}$ on the test data.

5.2. Comparison between aeroBERT-Classifier and Other Text Classification LMs

aeroBERT-Classifier was compared to other LMs capable of text classification with the performance metrics for each of the LMs summarized in Table 9. aeroBERT-Classifier and GPT-2 were fine-tuned and tested on aerospace requirements. Bi-LSTM (GloVe) was trained and tested on aerospace requirements from scratch. Lastly, an off-the-shelf ZSL classifier (bart-large-mnli) was used for classifying aerospace requirements without being trained/fine-tuned beforehand.

From Table 9, it is clear that despite the small requirements dataset used for fine-tuning/training the models, BERT, GPT-2, and Bi-LSTM, they outperform bart-large-mnli on in-domain text (aerospace requirements in this case). This underscores the importance of transfer-learning approaches in the aerospace requirements engineering domain, where the availability of labeled datasets is limited. The remainder of this section provides a one-on-one comparison between aeroBERT-Classifier$_{BC}$ and bart-large-mnli.

aeroBERT-Classifier is capable of classifying requirements into three types, as shown in Table 8. *bart-large-mnli*, on the other hand, is capable of classifying sentences into provided classes using NLI-based zero-shot Text Classification [56].

All the requirements present in the test set were classified using bart-large-mnli to facilitate the comparison with the aeroBERT-Classifier. The names of the types of requirements (Design Requirement, Functional Requirement, and Performance requirement) were provided to the model for zero-shot text classification.

Figure 11 shows the true and predicted labels for all the requirements in the test set. Upon comparing Figure 10 with Figure 11, aeroBERT-Classifier was able to correctly classify 83.87% of the requirements as compared to 43.39% when bart-large-mnli was used. The latter model seemed to be biased towards classifying most of the requirements as functional requirements. Had bart-large-mnli classified all the requirements as functional, the zero-shot classifier would have rightly classified 32.26% of the requirements. This illustrates the superior performance of the aeroBERT-Classifier despite it being trained on a small labeled dataset. Hence, while bart-large-mnli performs well on more general tasks such as sentiment analysis, classification of news articles into genres, etc., using zero-shot

classification, its performance is degraded in tasks involving specialized and structured texts such as aerospace requirements.

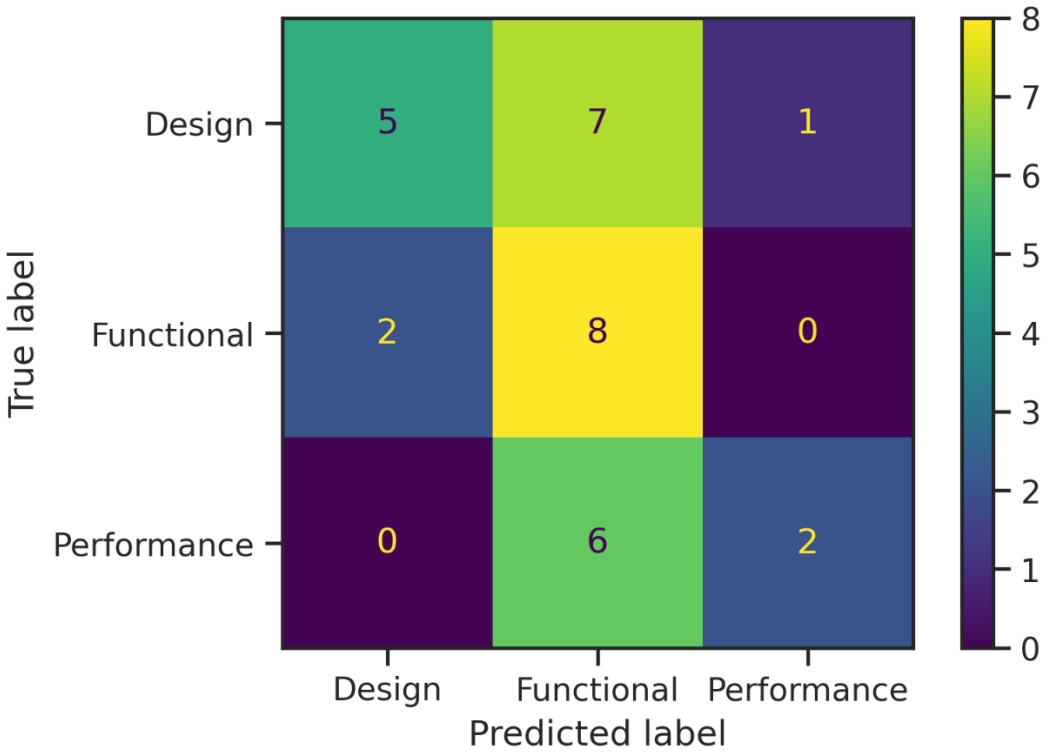

Figure 11. Confusion matrix showing the breakdown of the true and predicted labels by the bart-large-mnli model on the test data.

6. Conclusions and Future Work

The main contributions of this paper are the creation of an open-source classification corpus for aerospace requirements and the creation of an LM for classifying aerospace requirements. The corpus contains a total of 310 requirements along with their labels (Design, Functional, and Performance) and was used for fine-tuning the BERT LM to obtain aeroBERT-Classifier. A performance assessment of aeroBERT-Classifier achieved an average F1 score of 0.83 across all three requirement types in the unseen test data.

Finally, the aeroBERT-Classifier performed better at classifying requirements than GPT-2, Bi-LSTM (GloVe), and bart-large-mnli. This shows the benefits of using even small labeled datasets for fine-tuning pre-trained LMs such as BERT on downstream tasks in specialized domains.

Overall, aeroBERT-Classifier is a novel approach for classifying aerospace requirements. The results obtained support the fact that transfer learning can fuel aerospace requirements engineering research by paving the way for fine-tuning pre-trained language models with less training data while achieving generalizability.

One of the limitations of this work is that only three different types of requirements were considered. This was due to the fact that there were too few requirements of the other types to obtain a generalizable model for those classes. aeroBERT-Classifier was trained on requirements from Parts 23 and 25 of Title 14 CFRs, which are mainly certification requirements. Hence, while the model might not directly translate to the system requirements (which are proprietary) used by aerospace companies, the methodology demonstrated in this paper, however, remains applicable.

A logical next step to advance this work would be to include more requirements, and more types of requirements, to the corpus. Finally, comparing the performance of aeroBERT-Classifier against an LM trained on aerospace requirements from scratch and fine-tuned for the sequence classification task would be an interesting avenue for future exploration.

Author Contributions: Conceptualization, A.T.R., B.F.C., and O.J.P.F.; methodology, A.T.R. and R.T.W.; Software, A.T.R.; validation, A.T.R.; formal analysis, A.T.R.; investigation, A.T.R. and R.T.W.; data curation, A.T.R. and B.F.C.; writing—original draft preparation, A.T.R.; writing—review and editing, A.T.R., B.F.C., O.J.P.F., R.T.W., and D.N.M.; visualization, A.T.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The annotated requirements dataset created as a part of this work, can be found on the Hugging Face platform. Available online: https://huggingface.co/datasets/archanatikayatray/aeroBERT-classification (accessed on 1 February 2023).

Acknowledgments: The authors wish to thank the NVIDIA Applied Research Accelerator Program for hardware support. We would also like to thank the Hugging Face community for making BERT, GPT-2, and bart-large-mnli available, which were essential to this work.

Conflicts of Interest: The authors have declared that no competing interests exist.

Abbreviations

The following abbreviations are used in this manuscript:

BERT	Bidirectional Encoder Representations from Transformers
CFR	Code of Federal Regulations
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulations
GPT	Generated Pre-trained Transformer
INCOSE	International Council on Systems Engineering
LM	Language Model
LLM	Large Language Model
LSTM	Long Short-Term Memory
MBSE	Model-Based Systems Engineering
MISC	Miscellaneous
MNLI	Multi-Genre Natural Language Inference
NE	Named Entity
NER	Named Entity Recognition
NL	Natural Language
NLI	Natural Language Inference
NLP	Natural Language Processing
NLP4RE	Natural Language Processing for Requirements Engineering
QFD	Quality Function Deployment
RE	Requirements Engineering
SME	Subject Matter Expert
SOTA	State Of The Art
SysML	Systems Modeling Language
UML	Unified Modeling Language
ZSL	Zero-Shot learning

References

1. BKCASE Editorial Board. *Guide to the Systems Engineering Body of Knowledge*; INCOSE: San Diego, CA, USA, 2020; p. 945.
2. INCOSE. INCOSE Infrastructure Working Group Charter. pp. 3–5. Available online: https://www.incose.org/docs/default-source/working-groups/infrastructure-wg-documents/infrastructure_charter-ak-revision-3-feb-12-2019.pdf?sfvrsn=14c29bc6_0 (accessed on 10 January 2023).
3. NASA. Appendix C: How to Write a Good Requirement. pp. 115–119. Available online: https://www.nasa.gov/seh/appendix-c-how-to-write-a-good-requirement (accessed on 5 January 2023).
4. Firesmith, D. Are your requirements complete? *J. Object Technol.* **2005**, *4*, 27–44. [CrossRef]
5. NASA. 2.1 The Common Technical Processes and the SE Engine. Available online: https://www.nasa.gov/seh/2-1_technical-processes (accessed on 10 January 2023).
6. Nuseibeh, B.; Easterbrook, S. Requirements Engineering: A Roadmap. In Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland, 4–11 June 2000; Association for Computing Machinery: New York, NY, USA, 2000; pp. 35–46. [CrossRef]

7. Firesmith, D. Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them. *J. Object Technol.* **2007**, *6*, 17–33. [CrossRef]
8. Haskins, B.; Stecklein, J.; Dick, B.; Moroney, G.; Lovell, R.; Dabney, J. 8.4. 2 error cost escalation through the project life cycle. In *INCOSE International Symposium*; Wiley Online Library: Hoboken, NJ, USA, 2004; Volume 14, pp. 1723–1737.
9. Bell, T.E.; Thayer, T.A. Software requirements: Are they really a problem? In Proceedings of the 2nd International Conference on Software Engineering, San Francisco, CA, USA, 13–15 October 1976; pp. 61–68.
10. Dalpiaz, F.; Ferrari, A.; Franch, X.; Palomares, C. Natural language processing for requirements engineering: The best is yet to come. *IEEE Softw.* **2018**, *35*, 115–119. [CrossRef]
11. Ramos, A.L.; Ferreira, J.V.; Barceló, J. Model-based systems engineering: An emerging approach for modern systems. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2011**, *42*, 101–111. [CrossRef]
12. Estefan, J.A. Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group* **2007**, *25*, 1–12.
13. Jacobson, L.; Booch, J.R.G. *The Unified Modeling Language Reference Manual*; Addison-Wesley: Boston, MA, USA, 2021.
14. Ballard, M.; Peak, R.; Cimtalay, S.; Mavris, D.N. Bidirectional Text-to-Model Element Requirement Transformation. In Proceedings of the 2020 IEEE Aerospace Conference, Big Sky, MT, USA, 7–14 March 2020; pp. 1–14.
15. Lemazurier, L.; Chapurlat, V.; Grossetête, A. An MBSE approach to pass from requirements to functional architecture. *IFAC-PapersOnLine* **2017**, *50*, 7260–7265. [CrossRef]
16. BKCASE Editorial Board. *Needs, Requirements, Verification, Validation Lifecycle Manual*; INCOSE: San Diego, CA, USA, 2022; p. 457.
17. Wheatcraft, L.; Ryan, M.; Llorens, J.; Dick, J. The Need for an Information-based Approach for Requirement Development and Management. In Proceedings of the INCOSE International Symposium, Biarritz, France, 11–13 September 2019; Wiley Online Library: Hoboken, NJ, USA; Volume 29, pp. 1140–1157.
18. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 4171–4186. [CrossRef]
19. FAA. *Title 14 Code of Federal Regulations*; FAA: Washington, DC, USA, 2023.
20. Pennington, J.; Socher, R.; Manning, C. GloVe: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1532–1543. [CrossRef]
21. Ferrari, A.; Dell'Orletta, F.; Esuli, A.; Gervasi, V.; Gnesi, S. Natural Language Requirements Processing: A 4D Vision. *IEEE Softw.* **2017**, *34*, 28–35. [CrossRef]
22. Abbott, R.J.; Moorhead, D. Software requirements and specifications: A survey of needs and languages. *J. Syst. Softw.* **1981**, *2*, 297–316. [CrossRef]
23. Luisa, M.; Mariangela, F.; Pierluigi, N.I. Market research for requirements analysis using linguistic tools. *Requir. Eng.* **2004**, *9*, 40–56. [CrossRef]
24. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Baltimore, MD, USA, 23–24 June 2014; Association for Computational Linguistics: Baltimore, MD, USA, 2014; pp. 55–60. [CrossRef]
25. Natural Language Toolkit. Available online: https://www.nltk.org/ (accessed on 1 October 2022).
26. spaCy. Available online: https://spacy.io/ (accessed on 1 October 2022).
27. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.u.; Polosukhin, I. Attention is All you Need. In *Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
28. Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; Zettlemoyer, L. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv* **2019**, arXiv:1910.13461. https://doi.org/10.48550/arXiv.1910.13461
29. Zhao, L.; Alhoshan, W.; Ferrari, A.; Letsholo, K.J.; Ajagbe, M.A.; Chioasca, E.V.; Batista-Navarro, R.T. Natural language processing for requirements engineering: A systematic mapping study. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–41. [CrossRef]
30. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. In Proceedings of the NeurIPS *EMC*^{2} Workshop Vancouver BC, Canada, 13 December 2019.
31. Goldberg, Y. Neural network methods for natural language processing. *Synth. Lect. Hum. Lang. Technol.* **2017**, *10*, 1–309.
32. Jurafsky, D.; Martin, J.H. Speech and Language Processing (Draft). 2021. Available online: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf (accessed on 1 February 2023).
33. Niesler, T.R.; Woodland, P.C. A variable-length category-based n-gram language model. In Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings, Atlanta, GA, USA, 9 May 1996; IEEE: New York, NY, USA, 1996; Volume 1, pp. 164–167.
34. Bengio, Y.; Ducharme, R.; Vincent, P. A Neural Probabilistic Language Model. In *Advances in Neural Information Processing Systems*; Leen, T., Dietterich, T., Tresp, V., Eds.; MIT Press: Cambridge, MA, USA, 2000; Volume 13.

35. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. In Proceedings of the International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013.
36. Graves, A. Generating Sequences With Recurrent Neural Networks. *arXiv* **2013**, arXiv:1308.0850. https://doi.org/10.48550/arXiv.1308.0850.
37. Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the EMNLP, Doha, Qatar, 25–29 October 2014.
38. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA, USA, 7–9 May 2015.
39. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the *Advances in Neural Information Processing Systems 27 (NIPS 2014)*, Montreal, QC, Canada, 8–13 December 2014; Volume 27.
40. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
41. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*; Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H., Eds.; Curran Associates, Inc.: New York, NY, USA, 2020; Volume 33, pp. 1877–1901.
42. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
43. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to fine-tune bert for text classification? In Proceedings of the China National Conference on Chinese Computational Linguistics, Kunming, China, 18–20 October 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 194–206.
44. Alammar, J. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). Available online: https://jalammar.github.io/illustrated-bert/ (accessed on 1 February 2023).
45. Hey, T.; Keim, J.; Koziolek, A.; Tichy, W.F. NoRBERT: Transfer learning for requirements classification. In Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, 31 August–4 September 2020; IEEE: New York, NY, USA, 2020; pp. 169–179.
46. Dima, A.; Lukens, S.; Hodkiewicz, M.; Sexton, T.; Brundage, M.P. Adapting natural language processing for technical text. *Appl. AI Lett.* **2021**, *2*, e33. [CrossRef]
47. Sharir, O.; Peleg, B.; Shoham, Y. The cost of training nlp models: A concise overview. *arXiv* **2020**, arXiv:2004.08900.
48. Dai, A.M.; Le, Q.V. Semi-supervised sequence learning. In Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015; Volume 28.
49. Peters, M.E.; Ammar, W.; Bhagavatula, C.; Power, R. Semi-supervised sequence tagging with bidirectional language models. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); Association for Computational Linguistics: Vancouver, BC, Canada, 2017; pp. 1756–1765. [CrossRef]
50. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding with Unsupervised Learning. 2018. Available online: https://openai.com/research/language-unsupervised (accessed on 1 February 2023).
51. Howard, J.; Ruder, S. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*; Association for Computational Linguistics: Melbourne, Australia, 2018; pp. 328–339. [CrossRef]
52. Hugging Face. Available online: https://huggingface.co/ (accessed on 1 October 2022).
53. Alammar, J. The Illustrated Transformer. Available online: https://jalammar.github.io/illustrated-transformer/ (accessed on 1 October 2022).
54. Cleland-Huang, J.; Mazrouee, S.; Liguo, H.; Port, D. Nfr. Available online: https://doi.org/10.5281/zenodo.268542 (accessed on 1 October 2022).
55. Zero-Shot Learning in Modern NLP. Available online: https://joeddav.github.io/blog/2020/05/29/ZSL.html (accessed on 1 October 2022).
56. Yin, W.; Hay, J.; Roth, D. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 3914–3923. [CrossRef]
57. Williams, A.; Nangia, N.; Bowman, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*; Association for Computational Linguistics: New Orleans, LA, USA, 2018; p. 1112
58. Alhoshan, W.; Zhao, L.; Ferrari, A.; Letsholo, K.J. A Zero-Shot Learning Approach to Classifying Requirements: A Preliminary Study. In *Requirements Engineering: Foundation for Software Quality*; Gervasi, V., Vogelsang, A., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 52–59.
59. Tikayat Ray, A.; Pinon-Fischer, O.J.; Mavris, D.N.; White, R.T.; Cole, B.F. aeroBERT-NER: Named-Entity Recognition for Aerospace Requirements Engineering using BERT. In *AIAA SCITECH 2023 Forum*; American Institute of Aeronautics and Astronautics, Inc.: Reston, VA, USA, 2023. [CrossRef]

60. Beltagy, I.; Lo, K.; Cohan, A. SciBERT: A pretrained language model for scientific text. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; Association for Computational Linguistics: Cedarville, OH, USA, 2019; pp. 3615–3620. [CrossRef]
61. Araci, D. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv* **2019**, arXiv:1908.10063.
62. Lee, J.; Yoon, W.; Kim, S.; Kim, D.; Kim, S.; So, C.H.; Kang, J. BioBERT: A pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **2020**, *36*, 1234–1240. [CrossRef]
63. Alsentzer, E.; Murphy, J.R.; Boag, W.; Weng, W.H.; Jin, D.; Naumann, T.; McDermott, M. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*; Association for Computational Linguistics: Minneapolis, MN, USA, 2019; pp. 72–78. [CrossRef]
64. Lee, J.S.; Hsiang, J. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *World Pat. Inf.* **2020**, *61*, 101965. [CrossRef]
65. FAA. *Overview—Title 14 of the Code of Federal Regulations (14 CFR)*; FAA: Washington, DC, USA, 2013.
66. Fundamentals of Systems Engineering: Requirements Definition. Available online: https://ocw.mit.edu/courses/16-842-fundamentals-of-systems-engineering-fall-2015/7f2bc41156a04ecb94a6c04546f122af_MIT16_842F15_Ses2_Req.pdf (accessed on 1 October 2022).
67. Wheatcraft, L.S. Everything You Wanted to Know About Interfaces, but Were Afraid to Ask. Available online: https://reqexperts.com/wp-content/uploads/2016/04/Wheatcraft-Interfaces-061511.pdf (accessed on 1 February 2023).
68. Spacey, J. 11 Examples of Quality Requirements. Available online: https://simplicable.com/new/quality-requirements (accessed on 1 February 2023).
69. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. *arXiv* **2017**, arXiv:1711.05101. https://doi.org/10.48550/ARXIV.1711.05101.