aeroBERT-Classifier: Classification of Aerospace Requirements using BERT

Archana Tikayat Ray * , Bjorn F. Cole , Olivia J. Pinon Fischer * , Ryan T. White , Dimitri N. Mavris

Posted Date: 6 February 2023

doi: 10.20944/preprints202302.0077.v1

Keywords: Requirements Engineering; Natural Language Processing; NLP; BERT; Requirements Classification; Text Classification

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, Georgia, 30332
[2] Lockheed Martin Space, Littleton, Colorado, 80127
[3] Department of Mathematical Sciences, Florida Institute of Technology, Melbourne, Florida, 32901
* Correspondence: atr@gatech.edu

Abstract: Requirements are predominantly written in Natural Language (NL), which makes them accessible to stakeholders with varying degrees of experience, as compared to a model-based language which requires special training. However, despite its benefits, NL can introduce ambiguities and inconsistencies in requirements, which may eventually result in system quality degradation and system failure altogether. 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 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 the development of an annotated aerospace requirements corpus and its use to fine-tune $\text{BERT}_{\text{BASE-UNCASED}}$ to obtain aeroBERT-Classifier: a new language model for classifying aerospace requirements into design, functional, or performance requirements. A comparison between aeroBERT-Classifier and bart-large-mnli (zero-shot text classification) showcased the superior performance of aeroBERT-Classifier on classifying aerospace requirements despite being fine-tuned on a small labeled dataset.

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

1. Introduction

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, stand-alone (not grouped), and verifiable, and is deemed necessary for stakeholder acceptability" [1]. Requirements are the first step towards designing systems, products, and enterprises. As such, 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

 doi:10.20944/preprints202302.0077.v1

- **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

Requirements can be of different "types" such as functional, non-functional, design, performance, certification, etc., based on the system of interest [4]. This typing 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]. Various stakeholders contribute to the requirements generation process, such as consumers, regulatory bodies, contractors, component manufacturers, etc. [7].

Well-defined requirements and good requirements engineering practices can lead to successful systems. 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 [8], and have dramatic engineering and programmatic consequences when caught late in the product life cycle [9,10]. Because Natural Language (NL) is predominantly used to write requirements [11], requirements are commonly prone to ambiguities and inconsistencies. This in turn increases the likelihood of errors and issues in the way they are formulated, with the consequences that we know. As a result, a way to standardize requirements to reduce their inherent ambiguities is urgently needed.

1.2. Shift towards Model-based Systems Engineering

As mentioned before, most of the systems in the present-day world are complex and hence need a comprehensive approach to their design and development [12]. To accommodate this need, there has been a drive toward the development and use of the practice of Model-Based Systems Engineering (MBSE), where activities that support the system design process are accomplished using detailed models as compared to traditional document-based methods [13]. Models capture the requirements as well as the domain knowledge and make them accessible to all stakeholders [14,15]. While MBSE shows great promise, the ambiguities and inconsistencies inherent to NL requirements hinder their direct conversion to models [16]. The alternative, which would consist of 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 a MBSE environment. Doing so would also allow for the identification of errors or gaps in requirements during the requirements elicitation phase, leading to cost and time savings. Despite these benefits, there are no standard tools or methods for converting NL requirements into a machine-readable/semi-machine-readable.

1.3. Research focus and expected contributions

A first step towards standardizing requirements consists of 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*, 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. The field of Natural Language Processing (NLP) has shown promise when it comes to classifying text. However, text classification has mainly been limited to sentiment analysis, news genre analysis, movie review analysis, etc., and has not been applied in a focused way to engineering domains like aerospace. To address the aforementioned need, the research presented herein leverages a pre-trained language model, the Bidirectional Encoder Representational Transformer (BERT) [17], which is then fine-tuned on an original labeled dataset of aerospace requirements for requirements classification. The remainder of

 doi:10.20944/preprints202302.0077.v1

this paper is organized as follows: Section II presents the relevance of NLP in requirements engineering and discusses various language models for text classification. Section III identifies the gaps in the literature and summarizes the research goal. Section IV presents the technical approach taken to collect data, create a requirements corpus, and fine-tune $\text{BERT}_{\text{BASE-UNCASED}}$. Section V discusses the results and compares the results from the aeroBERT-Classifier with that of bart-large-mnli (zero-shot text classification). Lastly, Section VI summarizes the research conducted as part of this effort and discusses potential avenues for future work.

2. Background

2.1. Natural Language Processing for Requirements Engineering (NLP4RE)

Requirements are almost always written in NL [18] to make them accessible to different stakeholders. According to various surveys, NL was deemed to be the best way to express requirements [19], and 95% of 151 software companies surveyed revealed that they were using some form of NL to capture requirements [20]. 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 [19]. 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 [11]. The availability of NL libraries/toolkits (Stanford CoreNLP [21], NLTK [22], spaCy [23], etc.), and off-the-shelf transformer-based [24] pre-trained language models (LMs) (BERT [17], BART [25], 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 [26]. Figure 1 shows a clear increase in the number of published studies in NLP4RE over the years. This underlines the fact that NLP plays a crucial role in requirements engineering and will only get more important with time, given the availability of off-the-shelf language models.

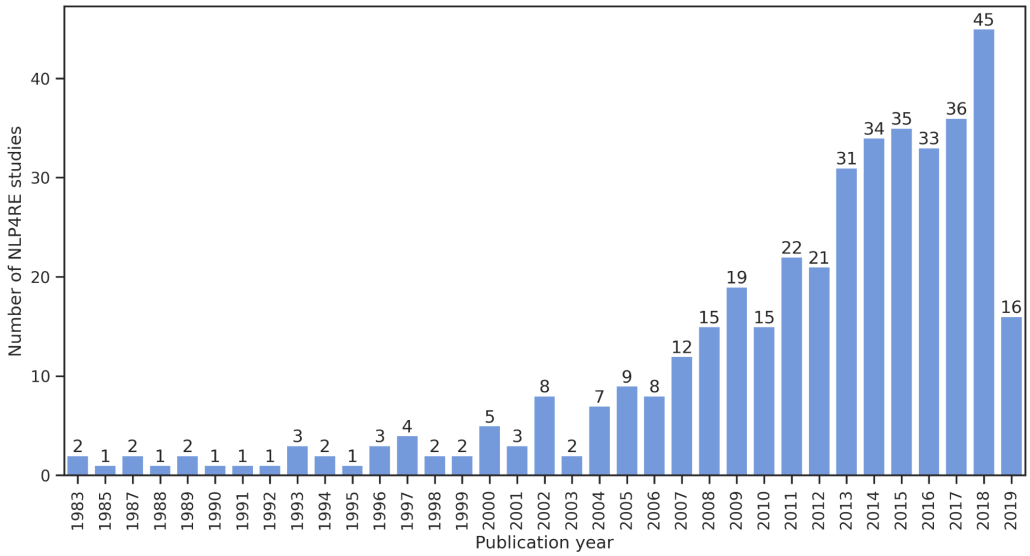

Figure 1. Published studies in the NLP4RE domain between 1983 and April, 2019 [26].

Among the 404 NLP4RE studies reported in [26], 370 NLP4RE studies were analyzed and classified based on the main NLP4RE task being performed. 26.22% of these studies focused on detecting linguistic errors in requirements (use of ambiguous phrases, conformance to boilerplates, etc.), 19.73%

 doi:10.20944/preprints202302.0077.v1

focused on requirements classification, and 17.03% on text extraction tasks focused on the identification of key domain concepts (Figure 2).

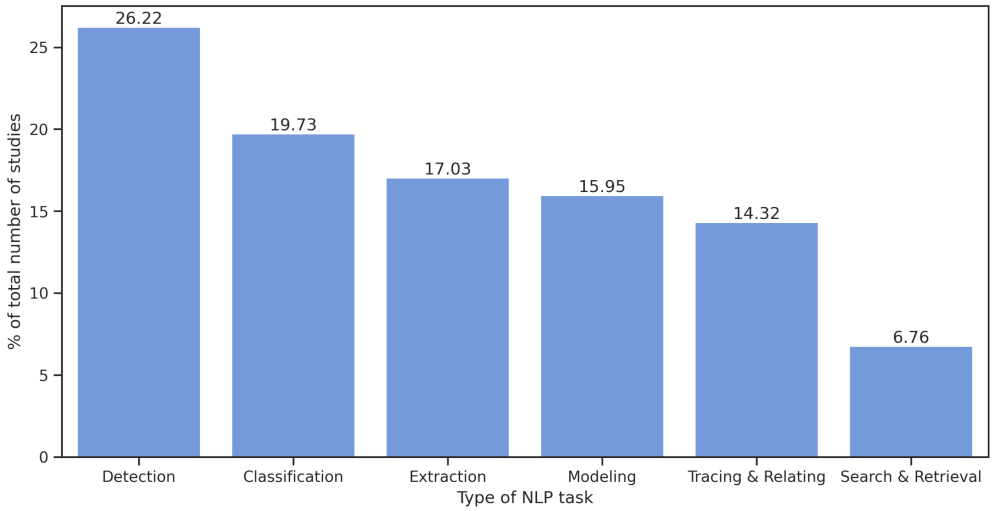

Figure 2. Distribution of different language tasks performed by studies surveyed by Zhao et al [26].

As mentioned, the classification of requirements is a critical step toward their conversion into a semi-machine-readable/standardized format. The following section discusses how NLP has evolved and can be leveraged to enable classification.

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

NLP is promising when it comes to classifying requirements, which is a potential step toward 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) [17,25,27]. Language modeling was classically defined as the task of predicting which word comes next [28]. Initially, this was limited to statistical language models [29], which use prior word sequences to compute the conditional probability for each of a vocabulary future word. The high-dimensional discrete language representations limit these models to N-grams [30], 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 2000s [31] and leveraged neural networks to simultaneously learn lower-dimensional word embeddings and learn to estimate conditional probabilities of 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 [32] to recurrent neural networks (RNNs) [33] and LSTM encoder-decoders [34] to attention mechanisms [35]. 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 beam search in [34] and sequence-to-sequence learning in [36]. These ideas were applied to distinct NLP tasks.

In 2017, the Transformer [24] 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 [24].

 doi:10.20944/preprints202302.0077.v1

All of these works required training on a single labeled dataset for a specific task. In 2018-20, 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 [17] (auto-encoding model), the Generative Pre-trained Transformer (GPT) family [37,38] of auto-regressive language models, and T5 character-level language model [39]. 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 (800M words) and English Wikipedia (2500M words) [17]. Similarly, GPT-3 is trained on 500B words gathered from datasets of books and the internet [38].

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 (Figure 3), such as Named-entity recognition (NER), text classification, language translation, question answering, etc. [17,40,41].

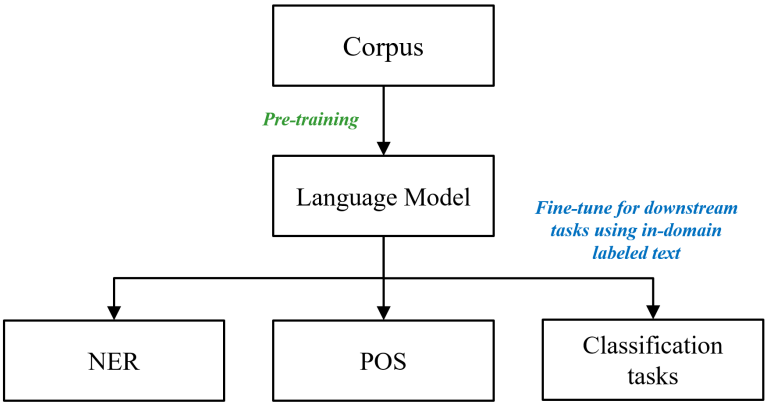

Figure 3. Chronology of Corpus, Language Model, and downstream tasks [42].

Training these models from scratch can be prohibitive given the computational power required: to put things into perspective, it took 355 GPU-years and $4.6 million to train GPT-3 [43]. Similarly, the pre-training cost for BERT$_{BASE}$ model with 110 million parameters varied between $2.5k-$50k, while it cost between $10k-$200k to pre-train BERT$_{LARGE}$ with 340 million parameters [44]. Pre-training the BART LM took between 12 days with 256 GPUs [25]. 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 [17,45–48].

Pre-trained LMs are readily available on the Hugging Face platform [49] 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 [17]. Figure 4 illustrates the pre-training and fine-tuning steps for BERT for a text classification task.

 doi:10.20944/preprints202302.0077.v1

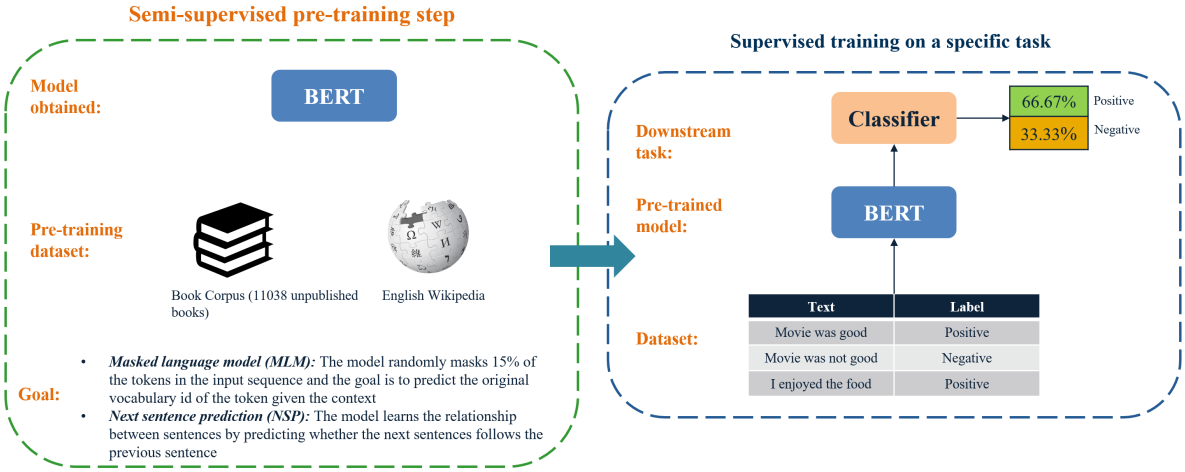

Figure 4. Fine-tuning a BERT language model on text classification task [17,50].

The following section provides details about some of the LMs used for text classification purposes.

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 [17]. 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 [24,50]. 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 [17]:

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

BERT is pre-trained on two self-supervised language tasks, namely, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) (Figure 4), 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 [51] to obtain NoRBERT (Non-functional and functional Requirements classification using BERT) - a model capable of classifying requirements [52]. 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 classification of the functional and quality aspect of requirements.

The PROMISE NFR dataset [51] contains 625 requirements in total (255 functional and 370 non-functional, which are further broken down into different "sub-types"). It is, to the authors' knowledge, the only requirements dataset of its kind that is publicly available. However, using this dataset was deemed implausible for this work because it predominantly focuses on software engineering systems and requirements. Table 1 shows some examples from the PROMISE NFR dataset.

 doi:10.20944/preprints202302.0077.v1

Table 1. Requirements examples from the PROMISE NFR dataset [51].

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

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 doing a task that they are not explicitly trained for is called zero-shot learning (ZSL) [53]. 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 [54]. The *bart-large-mnli* model was obtained by training *bart-large* [25] 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] [55]. 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 to probabilities associated with each of the labels provided.

Alhosan et al. [56] 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 uses a subset of the PROMISE NFR dataset as well [51].

The zero-shot learning method for classifiers has been introduced in this section to serve as a basis for LM performance in the classification task. The initial hypothesis behind this work is that the tuned aeroBERT-Classifier will outperform the bart-large-mnli model.

3. Research Gaps & Objectives

As stated previously, SOTA LMs 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 like 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 from 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 5. 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.

 doi:10.20944/preprints202302.0077.v1

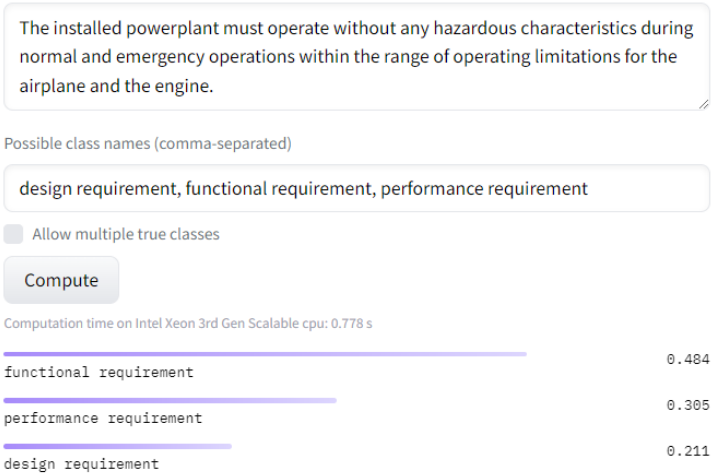

Figure 5. Classification of a performance requirement by bart-large-mnli model [54].

The existence of models such as *SciBERT* [57], *FinBERT* (Financial BERT) [58], *BioBERT* [59], *ClinicalBERT* [60], and *PatentBERT* [61] 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 the requirements classification task. In addition, because annotating RE 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 6 shows the pipeline consisting of different language models, the outputs of which put together will enable the conversion of NL requirements into semi-machine-readable requirements, which will inform boilerplate creation.

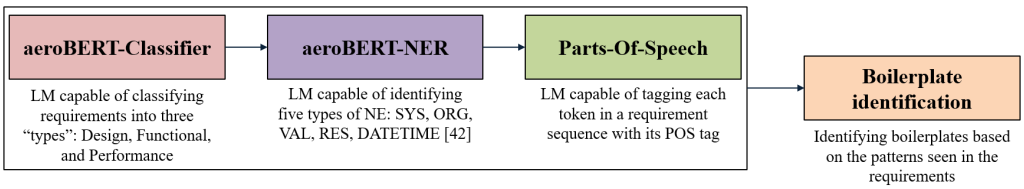

Figure 6. Pipeline for converting NL requirements to semi-machine-readable form enabling boilerplate creation.

This paper focuses on fine-tuning a language model for classifying requirements into various types. This effort achieves it through the:

1. **Creation of a labeled aerospace requirements corpus:** Aerospace requirements are collected from Parts 23 and 25 of Title 14 of the Code of Federal Regulations (CFRs) and annotated. The annotation involves tagging each requirement with its "type" (e.g., functional, performance, interface, design, etc.).
2. **Identification of aerospace requirements of interest:** There are different variations of the BERT LM for text classification, however, these models are trained on Tweets, sentiment data, movie reviews, etc. Hence, there is a need for a model that is capable of classifying requirements. Capabilities are added to identify three types of requirements: Design requirements, Functional requirements, and Performance requirements.

 doi:10.20944/preprints202302.0077.v1

3. **Fine-tuning of BERT for aerospace requirements classification:** The annotated aerospace requirements are used for fine-tuning $\text{BERT}_{\text{BASE-UNCASED}}$ LM. Metrics such as precision, recall, and F1 score are used to assess the model performance.

The following section describes in detail the technical approach taken.

4. Technical Approach

Aerospace requirements are often always proprietary, meaning that they are of limited open-source availability. Since large amounts of data are required to train a LM from scratch, this research focuses instead on *fine-tuning* the $\text{BERT}_{\text{BASE-UNCASED}}$ 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 7. The following sections describe each step in detail.

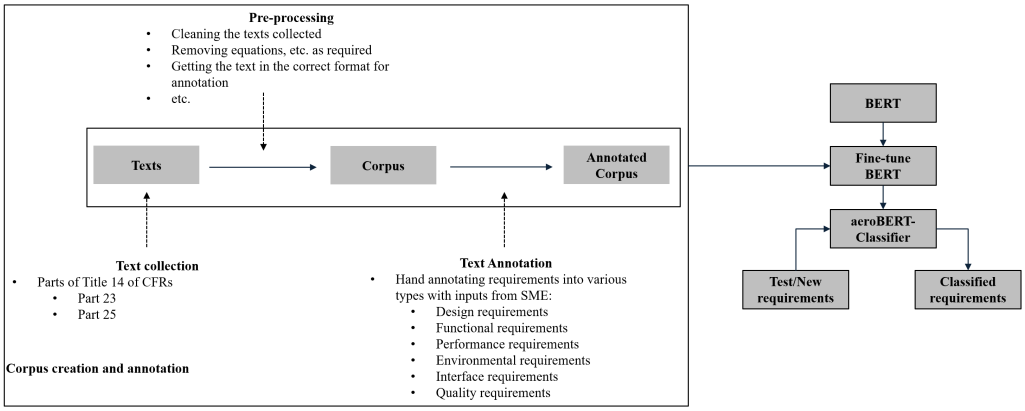

Figure 7. Methodology for obtaining aeroBERT-Classifier.

4.1. Data Collection, Cleaning, and Annotation

The creation of an annotated aerospace requirements corpus is a critical step since such corpus is not readily available in the open-source domain and is required for fine-tuning $\text{BERT}_{\text{BASE-UNCASED}}$. As seen in Figure 7, 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
1	Part 23: Airworthiness Standards: Normal, Utility, Acrobatic and Commuter Airplanes
2	Part 25: Airworthiness Standards: Transport Category Airplanes

[1] Texts were extracted from these resources and often had to be modified into requirements.

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.

 doi:10.20944/preprints202302.0077.v1

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 dynamic short period and Dutch roll stability in normal operations; and	**Requirement 2:** Airplanes not certified for aerobatics must have dynamic short 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 [42].

Table 4. Symbols that were modified for corpus creation [42].

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"/category of requirement. Requirements were classified into six categories, namely, Design, Functional, Performance, Interface, Environmental, and Quality. The definitions used for the requirement "types"/categories along with the 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.

 doi:10.20944/preprints202302.0077.v1

Table 5. Definitions used for labeling/annotating requirements [2,5,65].

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 flightcrew 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 [62]; **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 [63]; **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 8 shows the counts for all the requirement "types".

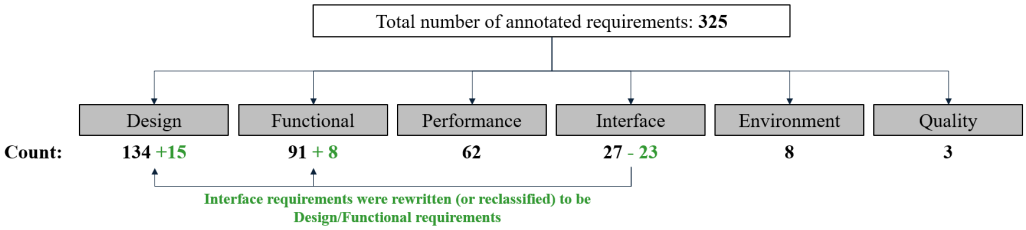

Figure 8. Six "types" of requirements were initially considered for the classification corpus. Due to the lack of sufficient examples for Interface, Environment, and Quality requirements, these classes were dropped at a later phase. However, some of the Interface requirements (23) were rewritten (or reclassified) to convert them into either Design or Functional requirements to keep them in the final corpus, which only contains Design, Functional, and Performance requirements.

As seen in Figure 8, the dataset is skewed toward Design, Functional, and Performance requirements (in that order). Since the goal is to develop a 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.

 doi:10.20944/preprints202302.0077.v1

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 8). 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 $BERT_{BASE\text{-}UNCASED}$

Language models (LMs) expect inputs in a certain format, and this may vary from one LM to another. BERT expects inputs in the format discussed below [17]:

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 9). 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 which are longer than 100, will be truncated.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 February 2023 doi:10.20944/preprints202302.0077.v1

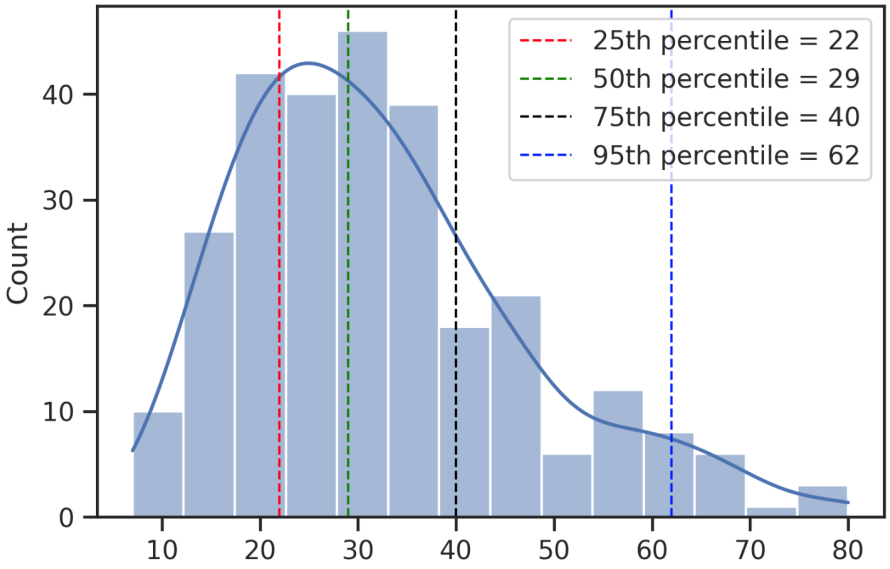

Figure 9. Figure showing the distribution of sequence lengths in the training set. The 95^{th} percentile was found to be 62. The maximum sequence length was set to 100 for the aeroBERT-Classifier model.

All the sequences were *lower-cased* since the casing is not significant when it comes to sequence classification. Preserving the casing is important for a token classification task such as Named-Entity Recognition (NER). The *bert-base-uncased* tokenizer was used for splitting the requirement sentences into WordPiece tokens, adding special tokens ([CLS], and [PAD]), mapping tokens to their respective IDs in $\text{BERT}_{\text{BASE-UNCASED}}$'s vocabulary of 30,522 tokens, padding/truncating sequences to match the maximum length, and the creation of attention masks (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 set. 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 test 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 $\text{BERT}_{\text{BASE-UNCASED}}$

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 [64] 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 were 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,

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 February 2023 doi:10.20944/preprints202302.0077.v1

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 seconds on an NVIDIA Quadro RTX 8000 GPU with a 40 GB VRAM. The small training time can be attributed to the small training set.

Figure 10 shows a rough schematic of the methodology used for fine-tuning the $BERT_{BASE\text{-}UNCASED}$ 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.

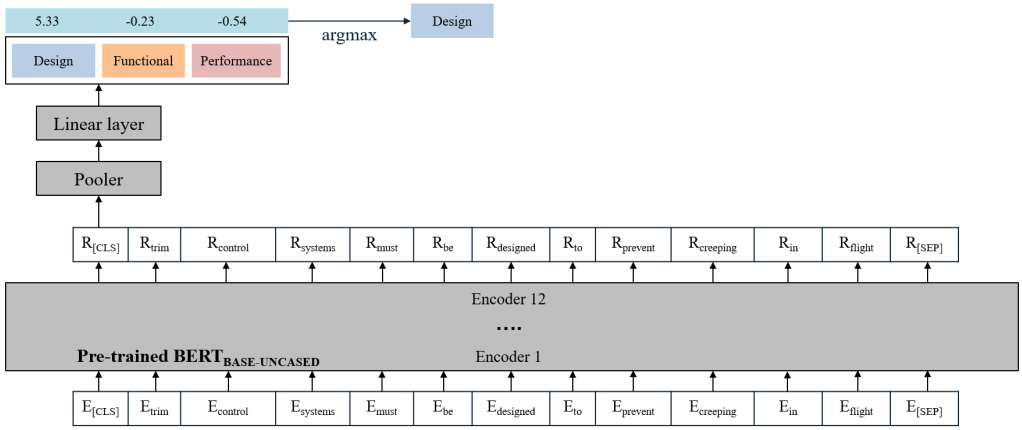

Figure 10. The detailed methodology used for full-fine-tuning of $BERT_{BASE\text{-}UNCASED}$ is shown here. E_{name} represents the embedding for that particular WordPiece token which is a combination of position, segment, and token embeddings. R_{name} is the representation for every token after it goes through the BERT model. Only $R_{[CLS]}$ is used for requirement classification since its hidden state contains the aggregate sequence representation [17].

The following section discusses the performance of aeroBERT-Classifier in more detail and compares its performance to that of bart-large-mnli on an aerospace requirements classification task.

5. Results

5.1. aeroBERT-Classifier Performance

aeroBERT-Classifier was trained on a dataset containing 279 requirements. The dataset was imbalanced, meaning there were more of one "type" of requirement as compared to others (136 design requirements compared to 89 functional, and 54 performance requirements). Therefore, precision, recall, and F1 score were chosen as the metrics for evaluating the model performance on the test set, as compared to *accuracy*. Table 9 provides the aggregate values for these metrics along with the breakdown for each requirement "type". The model was able to identify 92% (Recall) of design requirements present in the test set, however, of all the requirements that the model identified as design requirements, only 80% (Precision) belonged to this category. Similarly, the model 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 were 0.89 and 0.86, respectively.

 doi:10.20944/preprints202302.0077.v1

Table 9. Model performance for aeroBERT-Classifier on the test set.

Requirement type	Precision	Recall	F1 score
Design (0)	0.80	0.92	0.86
Functional (1)	0.89	0.80	0.84
Performance (2)	0.86	0.75	0.80
Average	**0.85**	**0.82**	**0.83**

The precision, recall, and F1 score for each requirement "types" were aggregated to obtain these scores for aeroBERT-Classifier and were found to be 0.85, 0.82, and 0.83, respectively. 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). A confusion matrix summarizing the classification task is shown in Figure 11. 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 (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 emergency operation of the airplane.	0	2
The microphone must be so located and, if necessary, the preamplifiers and filters of the recorder must be so 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 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 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 9). As seen in Table 10 and Figure 11, out of the 13 design requirements, only one was misclassified as a performance requirement. Of the 8 performance requirements, 2 were misclassified. And 2 of the 10 functional requirements were misclassified.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 6 February 2023 doi:10.20944/preprints202302.0077.v1

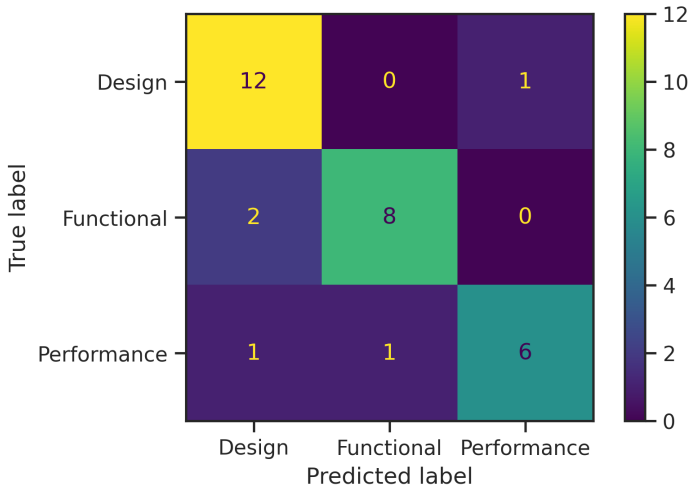

Figure 11. Confusion matrix showing the breakdown of the true and predicted labels by the aeroBERT-Classifier on the test data

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 an SME) who can make a judgment call on whether a certain requirement was labeled wrongly or to support a requirement rewrite to resolve ambiguities.

5.2. Comparison between aeroBERT-Classifier 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 [54].

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 12 shows the true and the predicted labels for all the requirements in the test set. Upon comparing Figure 11 to Figure 12, aeroBERT-Classifier was able to correctly classify 83.87% of the requirements as compared to 43.39% by bart-large-mnli. 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 being trained on a small labeled dataset. Hence, while bart-large-mnli performs well on more general tasks like 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.

 doi:10.20944/preprints202302.0077.v1

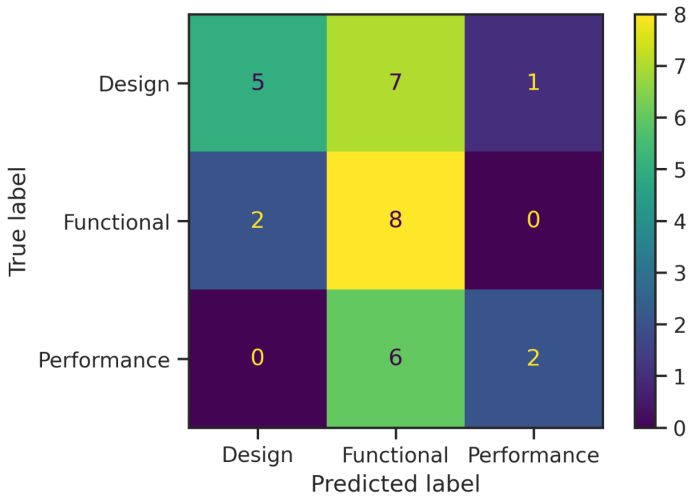

Figure 12. Confusion matrix showing the breakdown of the true and predicted labels by the bart-large-mnli model on the test data

6. Conclusions & 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 requirements. The corpus contains a total of 310 requirements along with their labels (Design, Functional, and Performance) and was used for fine-tuning a $\text{BERT}_{\text{BASE-UNCASED}}$ 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, aeroBERT-Classifier performed better at classifying requirements than bart-large-mnli. This shows the importance of using labeled datasets for fine-tuning LMs on downstream tasks in specialized domains.

A logical next step to advance this work would be to obtain more requirements to add to the corpus as well as more "types" of requirements. In addition, comparing the performance of aeroBERT-Classifier to a LM trained on aerospace requirements from scratch and fine-tuned for the sequence classification task would be an interesting avenue for further exploration.

7. Other details

Author Contributions:

- **Archana Tikayat Ray**: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing—original draft preparation, Writing—review and editing
- **Bjorn F. Cole**: Conceptualization, Data curation, Writing—review and editing
- **Olivia J. Pinon Fischer**: Conceptualization, Writing—review and editing
- **Ryan T. White**: Methodology, Writing—review and editing
- **Dimitri N. Mavris**: Writing—review and editing

Data Availability Statement: The annotated requirements dataset can be found on the Hugging Face platform. URL: https://huggingface.co/datasets/archanatikayatray/aeroBERT-classification.

Acknowledgments: The authors wish to thank NVIDIA Applied Research Accelerator Program for hardware support. We would also like to thank the Hugging Face community for making $\text{BERT}_{\text{BASE-UNCASED}}$ available, which was essential to this work.

 doi:10.20944/preprints202302.0077.v1

Abbreviations

Abbreviations

The following abbreviations are used in this manuscript:

BERT	Bidirectional Encoder Representations from Transformers
CFR	Code of Federal Regulations
COND	Condition
FAA	Federal Aviation Administration
FAR	Federal Aviation Regulations
GPT	Generated Pre-trained Transformer
INCOSE	International Council on Systems Engineering
LM	Language Model
LOC	Location (Entity label)
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
ORG	Organization (Entity label)
RE	Requirements Engineering
RES	Resource (Entity label)
SME	Subject Matter Expert
SOTA	State Of The Art
SYS	System (Entity label)
SysML	Systems Modeling Language
UML	Unified Modeling Language
ZSL	Zero-shot learning

References

1. Guide to the Systems Engineering Body of Knowledge; BKCASE Editorial Board, INCOSE, 2020; p. 945.
2. INCOSE. INCOSE INFRASTRUCTURE WORKING GROUP Charter **(accessed Jan. 10, 2023)**. pp. 3–5.
3. NASA. Appendix C: How to Write a Good Requirement **(accessed Jan. 05, 2022)**. pp. 115–119.
4. Firesmith, D. Are your requirements complete? *J. Object Technol.* **2005**, *4*, 27–44.
5. NASA. 2.1 The Common Technical Processes and the SE Engine. *J. Object Technol.* **(accessed Jan. 10, 2023)**, *4*.
6. Nuseibeh, B.; Easterbrook, S. Requirements Engineering: A Roadmap. Proceedings of the Conference on The Future of Software Engineering; Association for Computing Machinery: New York, NY, USA, 2000; ICSE '00, p. 35–46. doi:10.1145/336512.336523.
7. Regnell, B.; Svensson, R.B.; Wnuk, K. Can we beat the complexity of very large-scale requirements engineering? International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2008, pp. 123–128.
8. Firesmith, D. Common Requirements Problems, Their Negative Consequences, and the Industry Best Practices to Help Solve Them. *J. Object Technol.* **2007**, *6*, 17–33.
9. Haskins, B.; Stecklein, J.; Dick, B.; Moroney, G.; Lovell, R.; Dabney, J. 8.4. 2 error cost escalation through the project life cycle. INCOSE International Symposium. Wiley Online Library, 2004, Vol. 14, pp. 1723–1737.
10. Bell, T.E.; Thayer, T.A. Software requirements: Are they really a problem? Proceedings of the 2nd international conference on Software engineering, 1976, pp. 61–68.
11. Dalpiaz, F.; Ferrari, A.; Franch, X.; Palomares, C. Natural language processing for requirements engineering: The best is yet to come. *IEEE software* **2018**, *35*, 115–119.

12. Ramos, A.L.; Ferreira, J.V.; Barceló, J. Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **2011**, *42*, 101–111.
13. Estefan, J.A.; others. Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group* **2007**, *25*, 1–12.
14. Jacobson, L.; Booch, J.R.G. The unified modeling language reference manual **2021**.
15. Ballard, M.; Peak, R.; Cimtalay, S.; Mavris, D.N. Bidirectional Text-to-Model Element Requirement Transformation. *IEEE Aerospace Conference* **2020**, pp. 1–14.
16. Lemazurier, L.; Chapurlat, V.; Grossetête, A. An MBSE approach to pass from requirements to functional architecture. *IFAC-PapersOnLine* **2017**, *50*, 7260–7265.
17. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019, [arXiv:cs.CL/1810.04805].
18. Ferrari, A.; Dell'Orletta, F.; Esuli, A.; Gervasi, V.; Gnesi, S. Natural Language Requirements Processing: A 4D Vision. *IEEE Softw.* **2017**, *34*, 28–35.
19. Abbott, R.J.; Moorhead, D. Software requirements and specifications: A survey of needs and languages. *Journal of Systems and Software* **1981**, *2*, 297–316.
20. Luisa, M.; Mariangela, F.; Pierluigi, N.I. Market research for requirements analysis using linguistic tools. *Requirements Engineering* **2004**, *9*, 40–56.
21. Manning, C.; Surdeanu, M.; Bauer, J.; Finkel, J.; Bethard, S.; McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations; Association for Computational Linguistics: Baltimore, Maryland, 2014; pp. 55–60. doi:10.3115/v1/P14-5010.
22. Natural Language Toolkit. https://www.nltk.org/. (accessed: 01.10.2023).
23. spaCy. https://spacy.io/. (accessed: 01.10.2023).
24. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
25. 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. *CoRR* **2019**, *abs/1910.13461*, [1910.13461].
26. 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 Computing Surveys (CSUR)* **2021**, *54*, 1–41.
27. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* **2019**, *abs/1910.01108*.
28. Goldberg, Y. Neural network methods for natural language processing. *Synthesis lectures on human language technologies* **2017**, *10*, 1–309.
29. Jurafsky, D.; Martin, J.H. Speech and language processing (draft) **2021**.
30. Niesler, T.R.; Woodland, P.C. A variable-length category-based n-gram language model. 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings. IEEE, 1996, Vol. 1, pp. 164–167.
31. Bengio, Y.; Ducharme, R.; Vincent, P. A Neural Probabilistic Language Model. Advances in Neural Information Processing Systems; Leen, T.; Dietterich, T.; Tresp, V., Eds. MIT Press, 2000, Vol. 13.
32. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* **2013**.
33. Graves, A. Generating Sequences With Recurrent Neural Networks. [1308.0850v5].
34. Cho, K.; van Merrienboer, B.; Çaglar Gülçehre.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. EMNLP, 2014.
35. Bahdanau, D.; Cho, K.; Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; Bengio, Y.; LeCun, Y., Eds., 2015.
36. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. *Advances in neural information processing systems* **2014**, *27*.
37. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; others. Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.

38. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; Amodei, D. Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems; Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; Lin, H., Eds. Curran Associates, Inc., 2020, Vol. 33, pp. 1877–1901.
39. 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. *Journal of Machine Learning Research* **2020**, *21*, 1–67.
40. Sun, C.; Qiu, X.; Xu, Y.; Huang, X. How to fine-tune bert for text classification? China national conference on Chinese computational linguistics. Springer, 2019, pp. 194–206.
41. Alammar, J. The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning). https://jalammar.github.io/illustrated-bert/.
42. Ray, A.T.; 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*. doi:10.2514/6.2023-2583.
43. Dima, A.; Lukens, S.; Hodkiewicz, M.; Sexton, T.; Brundage, M.P. Adapting natural language processing for technical text. *Applied AI Letters* **2021**, *2*, e33.
44. Sharir, O.; Peleg, B.; Shoham, Y. The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900* **2020**.
45. Dai, A.M.; Le, Q.V. Semi-supervised sequence learning. *Advances in neural information processing systems* **2015**, *28*.
46. Peters, M.E.; Ammar, W.; Bhagavatula, C.; Power, R. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108* **2017**.
47. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving language understanding with unsupervised learning **2018**.
48. Howard, J.; Ruder, S. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146* **2018**.
49. Hugging Face. https://huggingface.co/. (accessed: 01.10.2023).
50. Alammar, J. The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer/.
51. Cleland-Huang, J.; Mazrouee, S.; Liguo, H.; Port, D. nfr. https://doi.org/10.5281/zenodo.268542.
52. Hey, T.; Keim, J.; Koziolek, A.; Tichy, W.F. NoRBERT: Transfer learning for requirements classification. 2020 IEEE 28th International Requirements Engineering Conference (RE). IEEE, 2020, pp. 169–179.
53. Zero-Shot Learning in Modern NLP. https://joeddav.github.io/blog/2020/05/29/ZSL.html. (accessed: 01.10.2023).
54. Yin, W.; Hay, J.; Roth, D. Benchmarking Zero-shot Text Classification: Datasets, Evaluation and Entailment Approach. *CoRR* **2019**, *abs/1909.00161*, [1909.00161].
55. Williams, A.; Nangia, N.; Bowman, S. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. 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, 2018, pp. 1112–1122.
56. Alhoshan, W.; Zhao, L.; Ferrari, A.; Letsholo, K.J. A Zero-Shot Learning Approach to Classifying Requirements: A Preliminary Study. Requirements Engineering: Foundation for Software Quality; Gervasi, V.; Vogelsang, A., Eds.; Springer International Publishing: Cham, 2022; pp. 52–59.
57. Beltagy, I.; Lo, K.; Cohan, A. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* **2019**.
58. Araci, D. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063* **2019**.
59. 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.
60. Alsentzer, E.; Murphy, J.R.; Boag, W.; Weng, W.H.; Jin, D.; Naumann, T.; McDermott, M. Publicly available clinical BERT embeddings. *arXiv preprint arXiv:1904.03323* **2019**.

 doi:10.20944/preprints202302.0077.v1

61. Lee, J.S.; Hsiang, J. Patentbert: Patent classification with fine-tuning a pre-trained bert model. *arXiv preprint arXiv:1906.02124* **2019**.
62. Wheatcraft, L.S. Everything you wanted to know about interfaces, but were afraid to ask. https://reqexperts.com/wp-content/uploads/2016/04/Wheatcraft-Interfaces-061511.pdf.
63. Spacey, J. 11 Examples of Quality Requirements. https://simplicable.com/new/quality-requirements.
64. Loshchilov, I.; Hutter, F., Decoupled Weight Decay Regularization; arXiv, 2017. doi:10.48550/ARXIV.1711.05101.
65. Fundamentals of Systems Engineering: Requirements Definition. Available online: 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).