# Deep Neural Networks in Natural Language Processing for Classifying Requirements by Origin and Functionality: An Application of BERT in System Requirements

**Article** *in* Journal of Mechanical Design · November 2023

DOI: 10.1115/1.4063764

CITATIONS
0

READS
53

**4 authors**, including:

Jesse Mullis
University of Georgia
**3** PUBLICATIONS **8** CITATIONS
SEE PROFILE

Cheng Chen
University of Georgia
**13** PUBLICATIONS **36** CITATIONS
SEE PROFILE

Beshoy Morkos
University of Georgia
**95** PUBLICATIONS **738** CITATIONS
SEE PROFILE

ASME Journal of Mechanical Design
Online journal at:
https://asmedigitalcollection.asme.org/mechanicaldesign

# Deep Neural Networks in Natural Language Processing for Classifying Requirements by Origin and Functionality: An Application of BERT in System Requirements

**Jesse Mullis**
College of Engineering,
University of Georgia,
302 East Campus Road,
Athens, GA 30602
e-mail: jessemm@uga.edu

**Cheng Chen**
College of Engineering,
University of Georgia,
302 East Campus Road,
Athens, GA 30602
e-mail: cheng.c@uga.edu

**Beshoy Morkos**[1]
College of Engineering,
University of Georgia,
302 East Campus Road,
Athens, GA 30602
e-mail: bmorkos@uga.edu

**Scott Ferguson**
Department of Mechanical and Aerospace Engineering,
North Carolina State University,
1840 Entrepreneur Drive,
Raleigh, NC 27606
e-mails: smfergu2@ncsu.edu;
scott_ferguson@ncsu.edu

*Given the foundational role of system requirements in design projects, designers can benefit from classifying, comparing, and observing connections between requirements. Manually undertaking these processes, however, can be laborious and time-consuming. Previous studies have employed Bidirectional Encoder Representations from Transformers (BERT), a state-of-the-art natural language processing (NLP) deep neural network model, to automatically analyze written requirements. Yet, it remains unclear whether BERT can sufficiently capture the nuances that differentiate requirements between and within design documents. This work evaluates BERT's performance on two requirement classification tasks (one inter- document and one intra-document) executed on a corpus of 1,303 requirements sourced from five system design projects. First, in the "parent document classification" task, a BERT model is fine-tuned to classify requirements according to their originating project. A separate BERT model is then fine-tuned on a "functional classification" task where each requirement is classified as either functional or nonfunctional. Our results also include a comparison with a baseline model, Word2Vec, and demonstrate that our model achieves higher classification accuracy. When evaluated on test sets, the former model receives a Matthews correlation coefficient (MCC) of 0.95, while the latter receives an MCC of 0.82, indicating BERT's ability to reliably distinguish requirements. This work then explores the application of BERT's representations, known as embeddings, to identify similar requirements and predict requirement change.*
[DOI: 10.1115/1.4063764]

*Keywords: requirement management, requirement classification, natural language processing, design automation, BERT*

## Introduction

Engineering designers rely on requirements to document, understand, and fulfill stakeholder needs. These requirements, encompassing both functional and nonfunctional aspects, are typically articulated in descriptive sentences within requirement documents. To elucidate the design's purpose and delineate relationships, these requirements are often structured hierarchically. This hierarchy is designed based on the company's culture and design practices. In many instances, the requirements are written in a hierarchical list with additional information such as origination date, responsible party, justification, verification method, and if changes have been made. Following the general framework of content-based recommendation systems [1], research in the field of requirement engineering seeks to create systems for requirement retrieval [2,3]. The challenge in finding relevant design information is emphasized by the tremendous amount of digitized data available across domains [4]. Consequently, automated procedures are preferred to laborious manual searches. Requirement engineers express a specific interest in the retrieval of requirements for their use in applications including requirement tracing [3], linking [5], reuse [6], and change prediction [7]. Addressing these requirements management challenges is crucial to project success, particularly when developing complex systems. This work seeks to investigate the efficacy of Bidirectional Encoder Representations from Transformers (BERT), a deep neural network, for representing requirements in a format

[1]Corresponding author.

Contributed by the Design Theory and Methodology Committee of ASME for publication in the Journal of Mechanical Design. Manuscript received June 29, 2023; final manuscript received October 2, 2023; published online November 13, 2023. Assoc. Editor: Christopher Mccomb.

  

suitable for automated analysis. The resulting representations have potential applications in requirements reuse and change prediction.

When designing new product variations that maintain the same solution principle, it is essential to reuse requirements, adapting only the embodiment to new constraints and requirements. In this design method, known as adaptive design, the functional structure can be modified by variation, addition, or omission since the general structure is well-known [8]. An adaptive design approach, commonly employed for complex, custom systems [9], necessitates the identification of similar requirements across different projects. For example, the Mars 2020 Perseverance Rover has many system requirements that overlap with its predecessor, the Mars Science Laboratory Curiosity Rover. Engineers at NASA JPL might utilize tools such as OpenCaesar to facilitate requirements reuse, as exemplified by the adaptive design of Perseverance from its predecessor Curiosity. But how can we leverage previous requirements when introducing a novel system design? This scenario was evident with the Small Robotic Helicopter (Ingenuity) that operated alongside Perseverance. In such situations, it is difficult to reuse requirements when no predecessor exists. To address this, one approach is to identify which existing system Ingenuity is most similar to and use that as a baseline for requirements. However, this is often difficult and subjective for engineers to do manually as there are many components and functions that overlap with multiple other systems. In such a scenario, it is helpful to use a tool to objectively determine which other system requirements the new system is most closely aligned to. Consider another example—Dyson's timely venture into designing ventilators during the pandemic. While Dyson doesn't produce ventilators, most of their products incorporate pneumatic mechanisms, so there is certainly some alignment with their existing product portfolio. Our approach could help Dyson identify which of their existing systems (e.g., Vacuum, Hair Dryer, Air Purifier) most closely aligns with the requirements of a ventilator. Such insights could guide decisions on personnel allocation, supplier considerations, and component selection to meet these requirements. With each project containing at minimum hundreds of requirements, designers may strain to identify similarities across the requirement corpus, leading to missed opportunities for reuse.

Though requirement elicitation occurs early in the design process, requirements often do not remain permanent throughout design projects. Requirement change propagation is a common phenomenon that occurs when an alteration to one requirement necessitates the subsequent change of other requirements [10,11]. Such change propagation can lead to unexpected project delays and expenses. Consequently, designers and stakeholders stand to benefit greatly from tools that can predict requirement change propagation ahead of time [12]. Previous research has shown that change propagation can be effectively predicted by the semantic and syntactic similarity of written requirements, where requirements' similarity is proportional to the likelihood of change propagating from one requirement to another [13]. However, previously proposed methods for measuring requirement similarity are computationally expensive and fail to capture a sufficiently detailed language representation. Advances in natural language processing (NLP) have led to the introduction of models, such as BERT, that could improve requirements management practices. Notably, BERT has proven its effectiveness in reducing ambiguities, noise, misspellings, and the prevalence of informal language in requirements [14].

BERT can project textual requirements into a continuous vector space conducive to computer analysis. The resulting vector representations are known as language embeddings [15]. Because of its utilization of transfer learning, BERT comes prepackaged with a highly developed understanding of natural language that can be fine-tuned to specific requirements management tasks. Previous studies have applied BERT to requirements [16–19], but it has yet to be shown whether BERT can reliably recognize variations in requirements that occur between industry projects. It is also unknown whether BERT can distinguish between types of requirements, i.e., functional requirements (FRs) and nonfunctional requirements (NFRs), within industry projects. Therefore, this work looks to answer the following research questions:

- RQ1: How may fine-tuned BERT's embedding of requirements be used to classify requirements based on source project?
- RQ2: How can the fine-tuned BERT embeddings of requirements be effectively utilized for accurate classification of FRs and NFRs in individual projects?
- RQ3: How effectively can BERT embeddings predict requirement change propagation?

Figure 1 displays a map of this research and indicates where each research question is addressed in the overall process. It is important to note that RQ1 and RQ2 are not designed to be exploratory [16,18,20]. Prior research has affirmed the capability of BERT models in executing classifications, given appropriate fine-tuning. This paper confirms that this outcome applies to requirements documents. We include the research questions to confirm that requirements exhibit similar classification behavior—which has not been studied previously. RQ1 and RQ2 ultimately support the pursuit of RQ3, which we consider to be the main contribution of the paper. The research questions are investigated using a dataset of 1303 requirements sourced from five mechanical design documents. Three of the documents come from private industry and detail the design of various manufacturing equipment, while the remaining two documents are publicly available and detail the design of subsystems for the Square Kilometer Array (SKA). The scope of this work only concerns analyzing the engineering requirement and non-requirements are not included in the dataset. To answer RQ1, a BERT model is fine-tuned for a "parent document classification (PDC)" task, where its objective is to classify requirements according to the project they came from. Requirement document reuse plays a vital role in the industry, providing a significant cost and time reduction for engineers during iterative design. Creating a new set of requirements can be time-consuming, and having a basis for the process can greatly enhance the efficiency of product development and mitigate potential errors. For example, automotive designers looking to create an Electric Vehicle (EV) could leverage existing requirement documents from gasoline vehicles. This would help identify components that can be borrowed from previous designs, which could potentially reduce delivery time and improve reliability for the new product. Then, to answer RQ2, a separate BERT model is fine-tuned for a "functional classification (FC)" task with the objective of classifying requirements as either FRs or NFRs.

After evaluating each model's performance, potential applications of the resulting requirement embeddings are explored; embeddings from the PDC BERT model are used to identify similar requirements both at the document and individual requirement

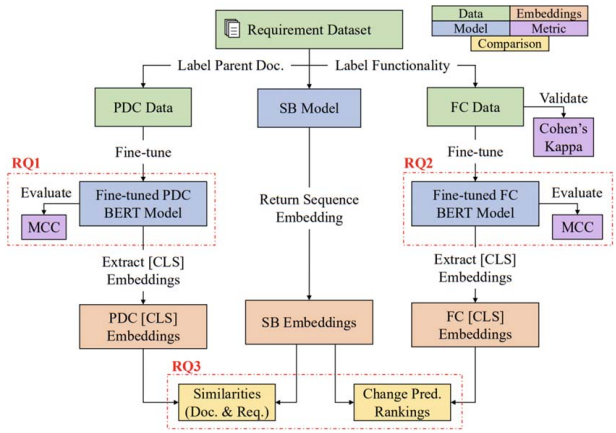


**Fig. 1 Research logic map**

level, while embeddings from the FC BERT model are applied to predict requirement change in two of the design projects. To answer RQ3, we utilize the combined embeddings (PDC and FC) to predict requirement change propagation and compare the results with those generated by a pretrained Sentence-BERT model. We attempt to discover if requirements, when classified in the proposed manners (PDC and FC) are more conducive to the accurate prediction of change propagation. For instance, can we utilize the insights gained from both the PDC and FC models to predict the propagation of requirement changes? The question can be addressed by computing the sentence similarity between a given sentence and the existing requirement subspace to estimate the most relevant change propagation. Beyond testing BERT's ability to sufficiently represent requirements, this work determines whether fine-tuning requirement classification yields higher-quality requirement embeddings than a general-language Sentence-BERT (SB) model.

## Background

In engineering settings, requirements are the purpose, goals, constraints, and criteria associated with a design project [13]. Requirements serve as the primary mode of communication and documentation of stakeholder needs and play a role in each stage of the design process, from project conception to completion. Unsurprisingly, they have been found to greatly impact a project's overall success [8,21]. Designers have consequently developed methods, such as those detailed in seminal design books to systematically elicit, compose, and manage requirements [8,22].

Though the particulars of requirement formats do, and indeed should [23,24], vary from industry to industry and project to project, requirements typically follow the same basic pattern. Designers most often express requirements as imperative statements containing the verbs "shall," "should," or "will." Appendix C in the NASA Systems Engineering Handbook [20] includes a checklist of recommendations for writing a good requirement. The checklist suggests, for example, that designers use "shall" for explicit requirements, "should" for goals, and "will" for facts or declarations of purpose; state requirements positively (e.g., use "shall" instead of "shall not") with correct grammar and spelling; convey one thought with a single subject and predicate. These recommendations describe a grammar for effective communication through requirements. Other guidelines, such as the "Guide for Writing Requirements" by the International Council on Systems Engineering [25] and "Simplified Technical English" by the Aerospace and Defense Industries Association of Europe [26], establish this grammar in further detail. The existence of these guides suggests the following:

(1) Requirements follow a set of rules that distinguish them from generic written text.
(2) Properly formed requirements are instrumental to the later requirements management process.

Based on these observations, a logical conclusion is that the representations used for requirement analysis should not be general-language representations but should instead be tailored specifically to requirements. It remains unclear whether modern NLP models, like BERT, can capture the nuances that distinguish requirements between and within engineering projects.

*Classifying Requirements.* Distinguishing between FRs and NFRs is a crucial step in requirements management. Not only do FRs appear earlier in the elicitation process than their nonfunctional counterparts [8], but they also pose different challenges and have varying effects on project success [27]. This paper follows Shankar's definition of FRs as "what a product must do, be able to perform, or should do" [28]. While there is broad agreement on this definition of FRs, there is no such consensus for the definition of NFRs [29]. Chung et al. attempt to define NFRs through a list of "-ilities" (e.g., compatibility, reliability) [29]. Other descriptors not ending in "ility," such as legal, operational, and security, can also be grouped under NFRs [30]. To maintain a binary, generalizable classification of requirements, this work considers NFRs to simply be any requirement that is not an FR.

The process of manually sorting requirements is time-consuming and labor-intensive, especially in complex systems with hundreds, if not thousands, of requirements. Several works attempt to automatically classify requirements with machine learning. Kurtanović and Maajel use a support vector machine model to classify requirements as FRs or NFRs using lexical features [30]. They use the PROMISE NFR dataset [31] containing 625 student-generated requirements mixed with user requirements extracted from online reviews. They achieve a precision and recall of around 92% for classifying FRs and NFRs and a precision of 93% and recall of 90% when further classifying NRFs into usability, security, operational, and performance requirements. Similar works apply other machine learning approaches to the PROMISE dataset, including convolutional neural networks (CNN) [20,32] and BERT [16]. Of the studies mentioned, a fine-tuned BERT model yields the best results, with an F1 score of 92%. These software requirements significantly differ from engineering ones, potentially limiting knowledge transfer to other domains. The dataset and subsequent research, primarily focused on software requirements, might not effectively extend to mechanical design requirements.

Akay and Kim bridge this gap between automated requirement classification and mechanical design by creating a synthetic dataset of mechanical design requirements, categorizing them as FRs or design parameters [33]. They then use BERT to create requirement embeddings that are fed to a support vector machine for classification. Though an impressive accuracy of 99.1% is achieved on a test set, the model is not validated on an authentic set of requirements. Therefore, a study is needed to verify the ability of advanced NLP models, like BERT, to classify a diverse set of requirements gathered from the industry. A workflow for fine-tuning BERT to classify industrial requirements into FRs and NFRs could later be applied to other types of requirement classification, such as groups based on subsystems (electrical, chemical, mechanical, software, etc.). Additionally, fine-tuning BERT for requirement classification tasks may yield requirement representations that could prove useful in other tasks, like identifying similar requirements and predicting requirement change.

*Identifying Similar Requirements.* Though there exists a need to identify similar requirements within adaptive and variant designs of mechanical products, most research investigating requirements similarity comes from the field of software engineering, where NLP approaches are applied to compute requirements' semantic similarity. Consequently, advances in requirements similarity research have largely coincided with advances in NLP. The framework proposed by Mihany et al. computes a similarity percentage for requirements documents based on shared words [6]. Other works explore the use of term frequency—inverse document frequency as well as latent semantic indexing [34,35]. Rajpathak et al. create a novel semantic similarity model that examines multi-phrase terms to identify "High," "Low," or "No Link" relationships between requirements [5]. More recent works employ neural network models to create links between software requirements. For example, Guo et al. applied two recurrent neural network (RNN) architectures, long short-term memory, and gated recurrent units, to replace previously state-of-the-art tracing methods [36]. The T-BERT framework then outperforms the RNN approach by applying a BERT model that has undergone both intermediate training and fine-tuning to associate natural language artifacts with corresponding programming language artifacts [18]. Abbas et al. evaluated the underlying assumption that semantic similarity relates to software similarity; they compared several different language models' evaluation of semantic similarity and found BERT's results to have the highest correlation with actual software similarity [19]. The BERT model used, though, is an SB model, trained to produce general sentence embeddings. Previous work has shown

that requirements documents differ from generic text in terms of structure and terminology [37], so a model trained to represent requirements specifically is preferred.

*Requirement Change Prediction.* In the context of mechanical design, an engineering change is defined as "an alteration made to parts, drawings or software that have already been released in the product design process" [38]. Engineering change notifications (ECNs) are the documents used to communicate and track the details of a change, such as its initiated date, cause, and the affected requirement [13]. While changes are inevitable and necessary to enhance designs or allow them to address new needs, they often come at the expense of delays and additional costs. Design projects are especially hampered by engineering change propagations, where implementing design changes results in the need for subsequent, unanticipated changes [39]. Designers have responded to this challenge by developing methods of predicting engineering change propagation.

Work by Morkos et al. implements change prediction at the requirement level using higher-order design structure matrices (DSMs) [40]. Higher-order DSMs track requirement relations beyond direct relationships. For example, if requirement "A" has a direct relationship with requirement "B," and requirement "B" has a direct relationship with requirement "C," then requirement "A" has a second-order relationship with requirement "C." Identifying relationships of order three and higher follows the same logic. The use of higher-order DSMs allows the model to predict change propagations that could not be identified by first-order relationships alone, which are the most obvious to human designers. A follow-up study compared three methods of identifying requirement relationships: manual, linguistic, and neural-network-based [13]. Though the presented linguistic approach provides the greatest accuracy in change prediction, it requires a labor-intensive process for tagging the desired parts of speech. Further work suggests that the accuracy of change prediction depends on whether requirements are linked through functional or nonfunctional relationships [41]. A model able to automatically represent semantic relationships while also distinguishing functional and nonfunctional relationships, such as BERT fine-tuned on the FC task, may present an opportunity for improved requirement change prediction.

**Bidirectional Encoder Representations From Transformers (BERT).** Upon its introduction, BERT demonstrated state-of-the-art results on various NLP tasks that include question answering, named-entity recognition, and, most important to this work, sequence classification [42]. BERT creates context-dependent vector representations of words, known as language embeddings. The original BERT model comes in two sizes: $\text{BERT}_{\text{BASE}}$, with 12 transformer encoder layers and 12 attention heads for a total of 110 M parameters, and $\text{BERT}_{\text{LARGE}}$, with 24 transformer encoder layers and 16 attention heads for a total of 340 M parameters. A problem with models of this scale is the shear amount of data and computational resources required for training. This obstacle is overcome through transfer learning, which involves training a model on a convenient task that improves the model's performance on a different task for which training is less convenient. In BERT's case, this involves pre-training on unsupervised tasks to develop a general-language understanding that can later be fine-tuned for a particular supervised task, which requires a labeled dataset. Specifically, BERT undergoes unsupervised pre-training on a "masked language model" task and "next sentence prediction" task applied to a corpus of 3300 M words. The masked language model task involves randomly replacing a word in an input sequence with a special token, designated "[MASK]" and then training the model to predict the replaced word. This task, adapted from the Cloze procedure [43], trains BERT to create word embeddings based on bidirectional context (i.e., based on both words to its left and its right), whereas previous transformer-based models were trained with a "next word prediction" task that could only consider a unidirectional context (i.e., words either to its right or to its left). The next sentence prediction task then involves inputting two sentences and having the model predict whether the two sentences occurred alongside one another in the source text. This task benefits BERT's performance when later fine-tuned to sentence-level tasks. Users also have the option to fine-tune the model with a custom dataset. When fine-tuned for sequence classification, BERT uses a special token, designated "[CLS]" (which stands for "classification") and appended to the beginning of each input sequence, to represent the whole sequence for an output classification layer. The [CLS] token embedding only contains useful information after the model undergoes fine-tuning; otherwise, the [CLS] token embedding cannot be considered an adequate representation of the entire sequence. Representations for entire sequences can also be generated using Sentence Transformers,[2] which trains BERT and other transformer-based models specifically to create sequence embeddings.

**Table 1 Dataset statistics**

| Project | Number of requirements | Avg. words per requirements | Vocabulary size |
|---|---|---|---|
| 1 | 350 | 19.7 | 1000 |
| 2 | 159 | 25.1 | 1379 |
| 3 | 214 | 27.6 | 1043 |
| 4 | 289 | 29.7 | 1342 |
| 5 | 291 | 40.6 | 1554 |
| ***Total*** | 1303 | 28.4 | 3765 |

## Research Methods

Three of the dataset's five requirements documents come from private industry and have undergone previous analyses [13,40,44–47]. In this study, we have proposed a heterogeneous dataset encompassing both similar and dissimilar requirements. Our goal is to create a repository that aids engineers in identifying reusable requirements across varying designs. This allows designers to efficiently filter out similar requirements and identify elements that could be beneficial for their new product design. Project 1 involves creating a production line for threaded pipes, Project 2 focuses on the design of manufacturing equipment for exhaust gas flaps, and Project 3 entails the design of industrial textile equipment. Projects 4 and 5 are publicly available requirements documents for designing subsystems of the SKA,[3] the world's largest radio telescope. Project 4 describes the design of a dish element, while Project 5 describes an artifact responsible for correlating and beamforming. Though this work does not claim this dataset to be a perfectly representative sample of all system requirements, the dataset seems diverse enough to provide, at the very least, an indication of BERT's ability to distinguish requirements in general. Table 1 displays statistics for each project's requirements document as well as statistics for the corpus as a whole. In total, the dataset contains 1303 requirements with an average length of 28.4 words and a vocabulary size (i.e., the number of unique words) of 3765.

**Creating Labeled Datasets.** Based on the demonstrated efficacy of BERT in multiple NLP tasks as well as its notable performance in specific requirement studies [14,18,42,48,49], we have chosen it as the centerpiece of our research investigation. To fine-tune BERT for PDC, each requirement must be labeled accordingly [50]. All requirements from Project 1, for example, are provided a label of "Doc1," and so on; providing these labels is trivial.

[2]https://www.sbert.net/index.html
[3]https://www.skatelescope.org/key-documents/

To create the dataset for FC fine-tuning, however, each requirement must be manually labeled as either "functional" or "nonfunctional." As covered in the background section, the distinction between these classes is not always clear, so one individual's labels may differ from another's. Therefore, some preliminary analysis is required to verify the assigned labels. Though the NFR definition is admittedly weak when compared to alternative definitions, it is used here to create a binary classification task that encompasses all requirements. The primary researcher developed a process to label all 1303 requirements in the dataset. Table 2 presents a breakdown of the labels. In total, around 70% of the dataset requirements are labeled as nonfunctional.

To verify these labels, a stratified, randomly generated, 10% sample of the dataset was labeled independently by two other research assistants. Before distributing requirements for labeling, we provided guidance and definitions of FRs and NFRs for annotating the labels. Building upon these generated labels, the incorporation of a random seed into later stratified random sampling improves the generalizability of training and testing data splits. By setting a fixed random seed, we ensure consistent sample assignments across different runs. This maintains the balance and representativeness of the selected samples in each split. Consistency in data partitioning aids in reliable model evaluation and enhances the validity and generalizability of findings.

Cohen's kappa statistic [51] is then used to evaluate inter-rater reliability between the primary researcher and each individual. Unlike percent agreement, the kappa statistic, $\kappa$, considers the possibility that raters may agree by chance. Equation (1) displays Cohen's kappa statistic formula, where $P(a)$ is the actual agreement among raters, and $P(e)$ is the expected agreement due to chance.

$$\kappa = \frac{P(a) - P(e)}{1 - P(e)} \tag{1}$$

Evaluating the first and second individual's labels against the primary researcher's labels results in Cohen's kappa statistics of 0.73 and 0.82, respectively. Following McHugh's interpretations [52], the computed statistics show a moderate and strong level of agreement, suggesting that the primary researcher's labels follow a coherent pattern. For any inconsistencies, we consulted with the system engineers responsible for designing the system to verify the accuracy of our labels. Additionally, the generated labels are checked by domain experts to ensure they are consistent with the design intent.

**Fine-Tuning for Requirement Classification.** Once labeled, the datasets are shuffled and split as follows: 80% of the requirements are allocated as a training set, and the remaining 20% are reserved as a test set. The training set is further broken down into 90% for training and 10% for validation. Similar to BERT's pretraining corpus, the requirement text does not undergo preprocessing (e.g., removal of stopwords) other than tokenization, where words are split into the "subtokens," which are the basic units of language that BERT represents with embeddings. Additionally, the class labels are converted into integers ranging from zero to [number_of_classes] −1 (e.g., a parent document label of "Doc 1" becomes "0"). This work uses the "bert-base-uncased" (letter casing is ignored) model in the Hugging Face [53] implementation of BERT for Sequence Classification. This model consists of 12 encoder layers, 12 self-attention heads, and an embedding size of 768. Fine-tuning the model for each requirement classification task occurs over three epochs of the 937 test set examples with a batch size of 16, AdamW optimizer with a learning rate of $2 \times 10^{-5}$, warmup ratio of 0.1, and weight decay of 0.01. During training, the model updates parameters to minimize cross-entropy loss, which maximizes the probability of correct classification. This procedure has a total runtime of 124 s for the PDC task and 97 s for the FC task when executed in a Google Colaboratory session equipped with a 16 GB NVIDIA P100 GPU. While the stated hyperparameters fall into the ranges recommended by BERT's creators, optimal values will vary depending on the dataset and may require adjustment when fine-tuning on a different set of requirements documents.

**Table 2 Label counts for FC task**

| Project | FRs | NFRs |
|---|---|---|
| 1 | 123 | 227 |
| 2 | 1 | 158 |
| 3 | 42 | 172 |
| 4 | 81 | 208 |
| 5 | 144 | 147 |
| ***Total*** | 391 | 912 |

To evaluate the proposed model's effectiveness, we benchmarked it against a Word2Vec baseline trained on partitioned data, using a minimum frequency count of 2 and a negative sampling rate of 10. After converting requirements into bigram representations with the Gensim library and compressing them to fixed-size vectors, the model was trained over 100 epochs. Subsequently, a RandomForest classifier was employed for predictions, and performance was assessed using precision and recall metrics.

*Evaluating Classification Performance.* This work relies on the MCC metric to evaluate the performance of the fine-tuned models. The MCC evaluates model performance across all classes, even with varying class sizes [54]. Given that both requirement classification tasks have an unbalanced dataset (i.e., varying number of samples in each class), the MCC is more informative than simple performance metrics, like accuracy. An MCC of one indicates perfect prediction, while an MCC of zero indicates a prediction equivalent to random guessing. Equation (2) displays the MCC formula for the binary classification case, where $x$ contains the true class labels, $y$ contains the predicted class labels, $Cov(x,y)$ is their covariance, and $t_p$, $t_n$, $f_p$, and $f_n$ are, respectively, the number of true positives, true negatives, false positives, and false negatives within $y$.

$$\mathrm{MCC_b} = \frac{Cov(x,\,y)}{\sigma_x \cdot \sigma_y} = \frac{t_p \cdot t_n - f_p \cdot f_n}{\sqrt{(t_p + f_p)(t_p + f_n)(t_n + f_p)(t_n + f_n)}} \tag{2}$$

Equation (2) indicates that the MCC is equivalent to measuring the correlation between $x$ and $y$ [55]; in fact, MCC is simply a special case of the Pearson correlation coefficient. The PDC task has five classes and requires the general multiclass calculation shown in Equ. 3, where $K$ is the number of classes, $s$ is the number of total samples, $c$ is the number of correctly predicted samples, $p_k$ is the number of times class $k$ was predicted, and $t_k$ is the number of times class k truly occurred.

$$\mathrm{MCC}_m = \frac{s \cdot c - \sum_{k=1}^{K} p_k \cdot t_k}{\sqrt{\left(s^2 - \sum_{k=1}^{K} p_k^2\right)\left(s^2 - \sum_{k=1}^{K} t_k^2\right)}} \tag{3}$$

It should be noted that while the MCC provides a useful, single-number evaluation of model performance, it does not capture all of the information contained in a confusion matrix, which contains counts of correct and incorrect predictions for each class. MCC provides an easily interpretable evaluation of performance, while the confusion matrix provides a more detailed, but less concise, performance evaluation.

**Obtaining and Analyzing Requirement Embeddings.** After fine-tuning a requirement classification task, BERT has learned requirement-specific embeddings that may prove useful in requirements management applications. This section explains the applied methodology for extracting and analyzing embeddings from

the fine-tuned models. The embedding analysis consists of a visualization of the embedding space followed by a numeric evaluation of requirement relationships.

*Extracting Embeddings From Fine-Tuned Models.* As previously covered, the [CLS] token embedding in BERT's last layer represents an entire input sequence prior to classification. To obtain this embedding, each requirement in the dataset is individually input into the fine-tuned model, which then outputs the hidden states corresponding with the requirement. The last layer (13th for $\text{BERT}_{\text{BASE}}$ since initial WordPiece embeddings are included) of the hidden states is extracted, and then, the first item of each sequence is selected (the [CLS] token is always appended to the beginning of each sequence). Following this procedure for each requirement yields a set of [CLS] embeddings for the entire dataset.

*Analyzing Requirement Relationships.* T-Distributed stochastic neighbor embedding (t-SNE) plots are used here to project the [CLS] vector, of dimension 768, into two-dimensional space for visualization. It is important to note that t-SNE plots do not preserve Euclidean distance between data; rather, the t-SNE algorithm minimizes a cost function equal to the Kullback–Leibler divergence of the joint probabilities of the original data and the two-dimensional projections. However, the cost function is not convex, so results may vary slightly for different initializations. For a detailed explanation of the t-SNE algorithm, see the original t-SNE paper by van der Maaten [56]. This work relies on the Scikit-Learn [57] implementation of t-SNE, which automatically applies principal component analysis, a dimensionality reduction technique, as a preprocessing step to improve the quality of the resulting embeddings. Hyperparameters, such as perplexity and learning rate, are kept at their default values.

Cosine similarity is commonly used to represent two vectors' relationship and is applied here to determine a requirement's nearest neighbors in the [CLS] embedding space. Results from a nearest neighbors search are easiest to interpret with positive-valued distances, so cosine similarity is converted to cosine distance. The conversion involves simply subtracting cosine similarity from one. Whereas cosine similarity is bounded by [−1, 1], cosine distance is always a positive number bounded by [0, 2]. Equation (4) presents the formula for cosine distance, where **A** and **B** are two vectors with angle, $\theta$, between them.

$$\text{cosine distance} = 1 - \cos(\theta) = 1 - \frac{\boldsymbol{A} \cdot \boldsymbol{B}}{||\boldsymbol{A}||\ ||\boldsymbol{B}||} \tag{4}$$

It should be noted that cosine similarity, and by extension cosine distance, are not "true" metrics since they do not satisfy the triangle inequality. However, cosine distance is preferred to alternatives like Euclidean distance for several reasons. First, cosine similarity is more commonly used within NLP. For example, the SB model uses a cosine similarity loss function when fine-tuned to compute semantic textual similarity [48]. Further, it has been shown that Euclidean distance becomes less meaningful in higher-dimensional spaces [58]. Lastly, cosine similarity's fixed boundaries make it more readily interpretable than unbounded distance metrics that could become inflated by arbitrary scaling.

**Identifying Similar Requirements.** Since the BERT model fine-tuned for PDC has developed [CLS] embeddings specifically designed to distinguish between requirements from different documents, these embeddings could provide a measure of the similarity between project requirements. The rationale here is that BERT will place similar requirements in similar areas in the [CLS] embedding space. There are no ground-truth similarity measures for this dataset, so the findings will be evaluated against intuition as well as results computed with a semantic textual similarity SB model. Similarity is observed at two levels: the document level and the individual requirement level. The document-level analysis could aid designers in evaluating the overlap between projects as a whole, while a requirement-level similarity search could identify specific opportunities for the reuse of design solutions.

The similarity amongst documents is explored by measuring the cosine distance between the average [CLS] embedding of each document's requirements [42]. Compared to other techniques like pooling, averaging the [CLS] embeddings often results in better outcomes. Averaging is often considered better than pooling in certain contexts due to its ability to capture a more comprehensive representation of the data. Pooling typically involves selecting a single value (such as the maximum or average) to summarize a set of values. Future study will explore these alternative methods and their potential implications to address the concern regarding the potential loss of important information as a result of average [CLS] embedding. Despite these potential drawbacks, the averaging approach is often used in practice due to its simplicity and efficiency. It provides a reasonable approximation of the document's overall representation and can still capture the general semantic meaning and relationships between requirements. Averaging the [CLS] embeddings for all requirements in a document produces a single, document-level embedding that is presumed to represent the entire project. To find similar requirements individually, the [CLS] embedding for a requirement of interest is input to the Facebook AI Similarity Search (Faiss) algorithm [59], which then searches among the set of [CLS] embeddings of all other requirements. Requirements with the same parent document as the requirement of interest are omitted from the search since the objective is to identify similar requirements in other projects that could be reused. Here, the cosine distance metric is used to find the three nearest neighbors to the query requirement. The Faiss algorithm is used because of its ability to efficiently search for nearest neighbors among dense, high-dimensional vectors. Though it does not directly support searches based on cosine similarity, Faiss does support inner product searches. By first normalizing the [CLS] embeddings according to their magnitude (i.e., placing them all on the unit sphere), the inner product search becomes equivalent to a cosine similarity search. Results are then presented in terms of cosine distance.

**Requirement Change Propagation.** In the background, it is noted that requirement change can be predicted by their semantic similarity. Changes propagate differently between FRs and NFRs. The [CLS] embeddings from the fine-tuned BERT for FC capture both semantic meaning and functionality of requirements. Using these embeddings, this study extends prior research, predicting change based on functional and nonfunctional links. ECNs from Project 3 test the predictive power of these embeddings. Initially affected requirements serve as a basis for forecasting subsequent ECN impacts. Their FC [CLS] embeddings are used as queries, and Faiss algorithm sorts the remaining requirements by cosine distance. The position in this sorted list is termed as its "ranking." Effective prediction means future-impacted requirements would have high ranks. Results are benchmarked against those using SB embeddings under the same ECNs.

**Obtaining SB Embeddings.** Though it may seem logical, the fine-tuned [CLS] embeddings cannot simply be compared to their values prior to fine-tuning since they do not meaningfully represent the input sequence at that stage. The Sentence Transformer models are based on work that trains BERT to create embeddings specifically for sequences of text, rather than individual tokens [48]. The model used here is "multi-qa-distilbert-cos-v1," which is built on DistilBERT (a smaller, distilled version of BERT) and fine-tuned on a dataset of 215 M question–answer pairs to identify, via cosine similarity, text relevant to a given query. Since both requirements management applications rely on textual similarity, this SB model should serve as a good baseline for judging the performance of the [CLS] embeddings. After embeddings have been computed, the output is normalized by the L2 norm to be used by downstream routines, such as the Faiss algorithm.

**Table 3 Confusion matrix for PDC BERT model applied to test set**

| | | Predicted Parent Document | | | | | |
|---|---|---|---|---|---|---|---|
| | | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 | Totals |
| Actual parent document | Doc1 | 69 | 0 | 0 | 1 | 0 | 70 |
| | Doc2 | 1 | 36 | 2 | 4 | 0 | 43 |
| | Doc3 | 2 | 0 | 30 | 0 | 0 | 32 |
| | Doc4 | 0 | 0 | 0 | 58 | 0 | 58 |
| | Doc5 | 0 | 0 | 0 | 1 | 57 | 58 |
| | ***Totals*** | 72 | 36 | 32 | 64 | 57 | 261 |

## Results: Word2Vec and BERT Fine-Tuned for Parent Document Classification

Tables 3 and 4 present confusion matrices showing the performance of Word2Vec and the fine-tuned model on the test set. The model's predicted parent document for each requirement is indicated along the matrix columns, while the actual parent documents are indicated along the rows. With perfect performance, the classified parent document would always match the actual parent document, and all off-diagonal values in the matrix would be equal to zero. The correctly classified requirements are indicated along the matrix diagonal in green, and the misclassifications, at off-diagonals, are indicated in red.

Table 5 summarizes this information by presenting the overall document classification recall and precision for each model. The model's classification performance on a particular document can be understood through precision and recall. Precision indicates, as a percentage, how often the model is correct when it predicts a specific class label. Precision can be calculated down a column by dividing the number highlighted in green by that column's total. Recall indicates, also as a percentage, how often a model can correctly predict the class label when given examples from a particular class. Recall can be calculated along a row by dividing the number highlighted in green by that row's total. The model has its lowest precision, 91%, for Document 4, and its lowest recall, 84%, for Document 2. Both Documents 2 and 5 have perfect precision, while Document 4 has perfect recall. The fine-tuned model's overall performance is evaluated with MCC, which is calculated to be 0.95. This score is associated with a very high correlation between predicted labels and actual labels [60] and suggests that the fine-tuned model can reliably distinguish requirements from different documents. We found that requirements are typically well-articulated, with strong internal correlations within each document due to shared thematic topics on component design. As illustrated in Fig. 2, some requirements correlate highly with sentences from other documents. Many such requirements pertain to functional aspects with specific technical details. The following requirements are chosen to demonstrate the semantic correlations when interpreting overlapping requirements:

**Table 4 Confusion matrix for PDC Word2Vec model applied to test set**

| | | Predicted Parent Document | | | | | |
|---|---|---|---|---|---|---|---|
| | | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 | Totals |
| Actual parent document | Doc1 | 36 | 0 | 2 | 1 | 4 | 43 |
| | Doc2 | 1 | 22 | 0 | 3 | 4 | 32 |
| | Doc3 | 0 | 0 | 51 | 0 | 7 | 58 |
| | Doc4 | 1 | 2 | 0 | 65 | 2 | 70 |
| | Doc5 | 1 | 2 | 2 | 0 | 53 | 58 |
| | ***Totals*** | 41 | 26 | 55 | 69 | 70 | 261 |

**Table 5 Precision and recall of Word2Vec and BERT across documents**

| | Word2Vec | | BERT | |
|---|---|---|---|---|
| | Precision | Recall | Precision | Recall |
| Doc1 | 0.878 | 0.837 | 0.958 | 0.986 |
| Doc2 | 0.846 | 0.688 | 1 | 0.837 |
| Doc3 | 0.927 | 0.879 | 0.938 | 0.938 |
| Doc4 | 0.942 | 0.929 | 0.906 | 1 |
| Doc5 | 0.779 | 0.914 | 1 | 0.983 |

(1) Provisions shall be provided to prevent mounting the wrong module (e.g., key coding of connectors).
(2) Each SKA1_Mid dish must simultaneously meet all requirements in both polarizations for at least the frequency ranges specified by SKA1-SYS_REQ-2180, 2181, 2182, 2183, 3612, and 3613.
(3) CSP_Mid.CBF should send an alarm message to CSP_Mid.LMC within 0.6 s once a measurement exceeds an alarm set-point.

While requirements 1 and 2 originate from the same document, requirements 2 and 3 exhibit notable similarity regarding "SKA1_Mid" or "CSP_Mid" system specifications. Despite their nuanced differences, this correlation is often stronger than with some requirements from the same document. Notably, both requirements 2 and 3 encompass the concept of time, a detail captured by the BERT model. Compared to the fine-tuned model, Word2Vec achieves a precision of 75% for Document 4 and a recall of 68% for Document 2. Document 4 has the highest scores with 92% recall and 94% precision. In contrast, the fine-tuned BERT model demonstrates higher classification prediction accuracy compared to the Word2Vec model. The accuracy may fluctuate depending on different random states (i.e., PYTHON hash seed environment variable values), yet the average accuracy remains relatively stable, leading to several requirements being assigned to different classes across simulations.

**Exploration of the Parent Document Classification Embedding Space.** Equipped with a model fine-tuned on the PDC task, [CLS] embeddings of all requirements in the dataset can now be retrieved from the model's final layer. The [CLS] embeddings are visualized as a t-SNE plot in Fig. 2, where requirements are colored according to their true parent document label. It is important to note that since the entire dataset is included, misclassifications among the training and validation sets are now evident.

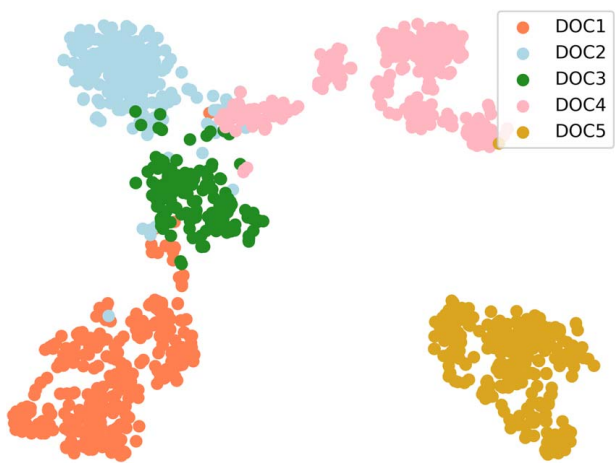


**Fig. 2 t-SNE Plot of [CLS] embeddings obtained from PDC task**

Despite misclassifications, the model manages to form a cluster for each document. The Document 1 cluster contains requirements from Documents 2 and 3, and the Document 2 cluster contains requirements from Documents 1 and 3. The 2D t-SNE plot was employed as a dimension reduction technique to visualize the multi-dimensional vector space in a 2D plane. Across documents, overlapping content refers to the use of similar words or semantics. The plot shows that certain clusters have significant overlap among the points. To interpret this, we can envision a centroid symbolizing each cluster in Fig. 2, where the centroid's position exhibits the smallest distance to every point within the cluster. The cosine similarity can be understood as a measure of the distance between these five centroids. A smaller cosine distance suggests a higher similarity or proximity between the clusters, indicating that the points within each cluster exhibit similar directions or orientations in vector space. For example, visually, the distance between documents 1 and 2 seems closer than 1 and 5. Consequently, the distance between 1 and 2 is smaller than that between 1 and 5 as shown in Table 6. Overall, Documents 1 and 5 appear to be the most distinct from one another, while Documents 3 and parts of Documents 2 and 4 have some overlap. According to the section headings in the original PDF version of Document 4, the overlapping requirements pertain to "Repair and Replacement," "Manufacturing Data Packs," "Packaging, Handling & Transportation," and "Safety and Security." These topics exhibited a significant overlap in terms of shared words and semantic meanings. These topics contrast those seen in the rest of Document 4, which is mainly composed of technical specifications related to the function and operation of the SKA's dish element. Conversely, these topics seem to overlap with those frequently found in Documents 1, 2, and 3, which all detail the production of manufacturing equipment.

*Comparison with SB Embedding Space.* The SB embedding space is explored here to make evident the differences between the parent document embedding space and that of a model trained to create embeddings for general text sequences. Note that the document embedding (PDC and FC) does not contain any non-requirement information. Figure 3 displays a t-SNE plot of the complete dataset using embeddings generated by the SB model. Documents appear clustered, though not as distinctly as they are in Fig. 2. Documents 1, 2, and 3 form a larger cluster, which includes some requirements from Document 4 as well. Once again, Document 5 emerges as the most distinct document. In all, the SB embeddings do appear to capture patterns that differentiate requirements between documents, but to a lesser degree than the [CLS] embeddings extracted from the PDC model.

**Document-Level Similarity.** Representations for documents are obtained by averaging the [CLS] embeddings for all requirements in a document [42]. A cosine distance matrix is shown in Table 6 for the averaged embeddings of all five documents. Documents 2 and 3 are separated by the smallest cosine distance of 0.62, while Document 1 is the next nearest to Documents 2 and 3 with cosine distances of 0.79 and 0.65, respectively. Documents 3 and 4 have the closest remaining relationship with a cosine distance of 0.86. Document 5 is the most distinct, with cosine distances of at least 1.0 separating it from all other documents. Documents 1 and 5 are separated by the most significant observed distance of 1.39.

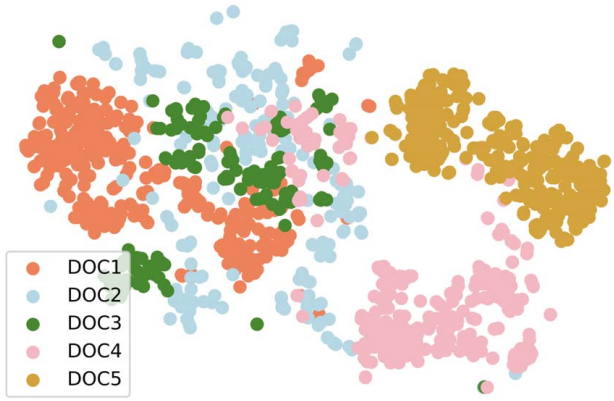


**Fig. 3 t-SNE Plot of project requirements in SB embedding space**

While there is no ground-truth to evaluate the results against, there does appear to be sound backing for a couple of high-level observations. First, Projects 1, 2, and 3 are closer to one another than they are with Projects 4 and 5. This relative proximity is expected since Projects 1, 2, and 3 all describe the design of manufacturing equipment. Next, even though Document 5 is widely separated from all documents, it is closer to Document 4 than it is to Documents 1, 2, and 3, which makes sense given that Documents 4 and 5 originate from the SKA project. It is surprising, however, that despite coming from the same project, Documents 4 and 5 are separated by a cosine distance of 1.00. A potential explanation for such a large distance is that individual subsystems may have little relation or interaction in the design of a system as immense and complex as the SKA. In fact, Document 4 makes no mention of the correlator and beamformer described in Document 5. Searching Document 5 for the dish element described in Document 4 reveals that it was mentioned in only 4 of the 289 requirements. This procedure for computing document similarity appears to reflect logical relationships between requirements documents while also unveiling similarities that may not be immediately obvious to a designer reviewing documentation.

*Comparison to SB Results.* The same document-level similarity analysis is performed with the SB embeddings for comparison. The cosine distance matrix for the documents' averaged embeddings is displayed in Table 7. The cosine distances computed with the SB embeddings are generally lower than those computed with the PDC model's [CLS] embeddings. The smallest cosine distance, with a value of 0.26, is observed between Documents 1 and 2 as well as Documents 2 and 3. Documents 1 and 3 are the next closest, with a cosine distance of 0.26. Document 5 is once again the most distinct and is separated from Documents 1, 2, and 3 by roughly the same amount, with cosine distances of 0.72, 0.73, and 0.70, respectively.

The general trends observed in Table 7 remain mostly consistent with those observed in Table 6. Documents 1, 2, and 3 are once again closer to each other than to Documents 4 and 5. Additionally,

**Table 6 Cosine distance between averaged PDC [CLS] embeddings**

| | Doc1 | Doc2 | Doc 3 | Doc4 | Doc5 |
|---|---|---|---|---|---|
| Doc1 | 0 | 0.79 | 0.65 | 1.02 | 1.39 |
| Doc2 | 0.79 | 0 | 0.62 | 0.88 | 1.16 |
| Doc3 | 0.65 | 0.62 | 0 | 0.86 | 1.22 |
| Doc4 | 1.02 | 0.88 | 0.86 | 0 | 1.00 |
| Doc5 | 1.39 | 1.16 | 1.22 | 1.00 | 0 |

**Table 7 Cosine distance between averaged SB embeddings**

| | Doc1 | Doc2 | Doc 3 | Doc4 | Doc5 |
|---|---|---|---|---|---|
| Doc1 | 0 | 0.26 | 0.32 | 0.58 | 0.72 |
| Doc2 | 0.26 | 0 | 0.26 | 0.42 | 0.73 |
| Doc3 | 0.32 | 0.26 | 0 | 0.57 | 0.70 |
| Doc4 | 0.58 | 0.42 | 0.57 | 0 | 0.52 |
| Doc5 | 0.72 | 0.73 | 0.70 | 0.52 | 0 |

Document 4 is the closest of all documents to Document 5. Some departures from the previously observed patterns are evident, such as Document 2, rather than Document 3, being closest to Document 4. While the SB embeddings do appear to capture many of the same document relationships as the PDC [CLS] embeddings, they generally yield less distinction between documents.

**Requirement-Level Similarity.** In this section, the PDC [CLS] embeddings are applied to identify similar requirements individually. Figure 4 demonstrates the search pipeline with an example search from the dataset. The input requirement in the example specifies a safety feature for ease of lubrication. The search algorithm returns three requirements from Document 2. With a cosine distance of 0.36, the nearest requirement declares that wiring must satisfy the relevant safety regulations. The following result, with a cosine distance of 0.38, states that perishable tooling should be easy to remove and install. Also with a cosine distance of 0.38, the final result states that equipment should have an ergonomic design. Ergonomics is typically considered a subcategory of safety, with its purpose being to reduce injuries that may develop over long periods of work. As indicated by the highlighted text in Fig. 4, the first and third results relate to the safety aspect of the input requirement, while the second result relates to the input through the concepts of removal and ease of maintenance.

This example demonstrates this procedure's ability to search multiple documents for related requirements. The relevance of some results is not always initially obvious, but further inspection can reveal relationships that designers may find useful. However, requirements are sometimes unique to their project, and searches may fail to produce relevant results at all. Additionally, there could be similar requirements in other documents that are not considered simply because those documents are further from the input requirement's parent document. For example, all three results in Fig. 4 come from Project 2; it could also be the case that only Project 2 requirements are returned simply due to Document 2's proximity to Document 3.

*Comparison to SB Results.* Using the same input requirement, a similarity search is conducted via the SB embeddings instead of the PDC [CLS] embeddings; the search results are displayed in Fig. 5. The first two results relate explicitly to safety, with the first indicating the need to mark hazardous equipment appropriately and the second specifying a particular safety regulation that must be satisfied. The third result comes from Document 2 and was also returned in the PDC [CLS] embedding search. The results as a whole have a greater range of cosine distances from the input than those shown in Fig. 4, suggesting that the SB embeddings represent individual requirements more distinctly than the PDC [CLS] embeddings. The SB embedding results also include requirements from two documents as opposed to a single document, which may indicate a more balanced search across the entire dataset. Overall, the two embedding types return comparable results in this case, but the SB embeddings consider requirement similarity alone, while the PDC [CLS] embeddings appear to return similar requirements from similar documents.

## Results: Bert Fine-Tuned for Functional Classification

The confusion matrix displayed in Tables 8 and 9 indicates the fine-tuned model's and Word2Vec's performance on the test set.

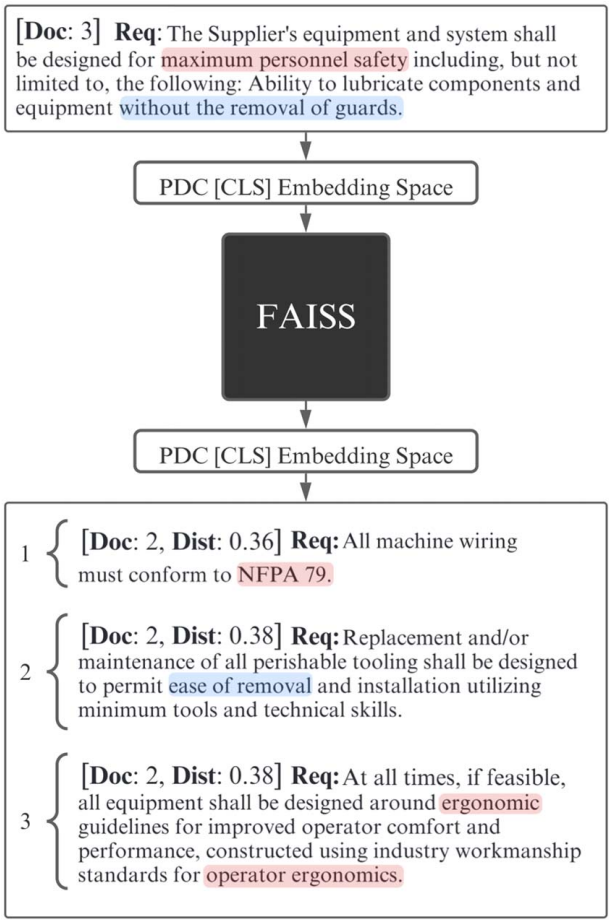


**Fig. 4 Requirement similarity search in PDC [CLS] embedding space**

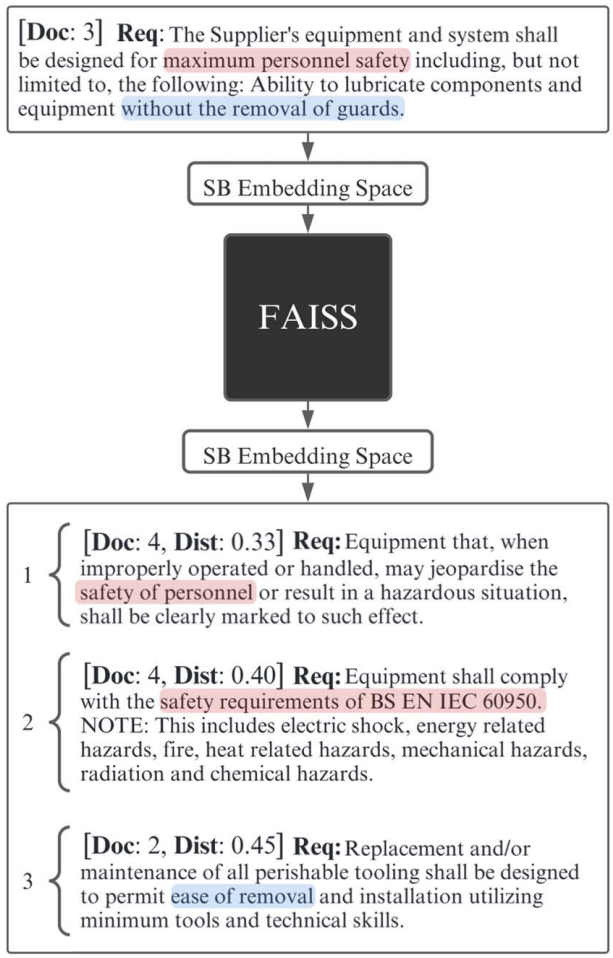


**Fig. 5 Requirement similarity search in SB embedding space**

**Table 8 Confusion matrix for FC BERT model applied to test set**

| | | Predicted label | | |
|---|---|---|---|---|
| | | Nonfunctional | Functional | **Totals** |
| Actual label | Nonfunctional | 179 | 6 | 185 |
| | Functional | 13 | 63 | 76 |
| | **Totals** | 192 | 69 | 261 |

**Table 9 Confusion matrix for FC Word2Vec model applied to test set**

| | | Predicted Label | | |
|---|---|---|---|---|
| | | Nonfunctional | Functional | **Totals** |
| Actual Label | Nonfunctional | 177 | 6 | 183 |
| | Functional | 19 | 59 | 78 |
| | **Totals** | 196 | 65 | 261 |

As a reminder, counts for the model's classified requirement labels are indicated along the matrix columns, and counts for the actual labels are indicated along the rows. A similar performance was achieved by both baseline model Word2Vec and fine-tuned BERT, as shown in Tables 3 and 4. Out of the 185 NFRs, the model correctly labeled 179 for a recall of 97%; out of the 76 FRs, the model correctly labeled 63 for a recall of 83%. In terms of precision, the model receives scores of 93% for NFRs and 91% for FRs. The overall performance is summarized by an MCC of 0.82, indicating a high correlation between the predicted labels and the actual labels [60]. Domain experts' assessments of MCC and T-SNE visualizations, depicted in Figs. 4 and 5, provide valuable insights into document similarity and verify each change propagation path. Evaluating requirement change depths and rankings with the actual change path enhances predictive understanding of change propagation. These evaluations, combined with t-SNE's visual representations, reveal change patterns and different types of relationships within requirement documents. To improve the predictive accuracy of NonFunctional Requirements (NFRs), unsupervised language learning models like ELMo, GPT, and other transformer-based models can be employed. These models provide distinctive advantages, including learning from unlabeled data, identifying hidden patterns, and managing out-of-vocabulary words. They allow for pre-training and transfer learning, capturing domain-specific knowledge representations, and extrapolating knowledge from extensive text datasets.

**Exploration of the Functional Classification Embedding Space.** The [CLS] embeddings are extracted from the FC BERT model and visualized in a t-SNE plot shown in Fig. 6. FRs and NFRs form distinct clusters, with a thin bridge connecting the two. The FR cluster could be considered two smaller clusters, while the NFRs are one sprawling cluster.

*Comparison with SB Embedding Space.* For comparison, a t-SNE plot for the SB embeddings is shown in Fig. 7. Note that this t-SNE plot is equivalent to the one shown in Fig. 3, except the coloring is changed to indicate FRs and NFRs rather than documents. The plot shows no clear distinction between NFRs, which span the embedding space, and FRs that exist in disconnected groups. The concentration of FRs could be due to relying on a specific definition for FRs but not for NFRs; only requirements that describe functionality are labeled as FRs, while all other requirements are lumped together as NFRs. Overall, any recognizable requirement clusters in the SB embedding space appear more indicative of parent document relationships than functional ones.

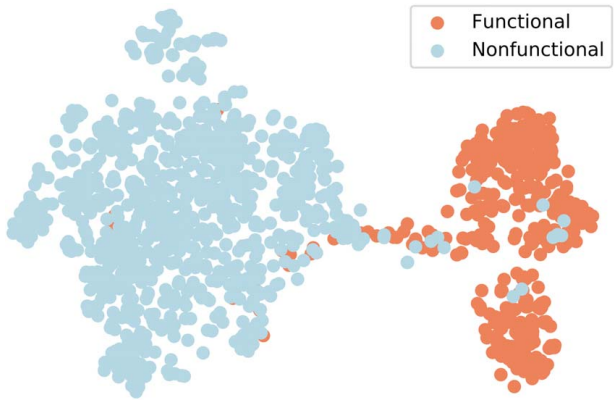


**Fig. 6 t-SNE Plot of [CLS] embeddings obtained from PDC task**


**Fig. 7 t-SNE Plot of FRs and NFRs in SB Embedding Space**

**Predicting Change in Project 3.** The application of requirement embeddings for change prediction is explored in Project 3. For the sake of brevity, specific requirements are referred to in the following sections as "R" followed by their requirement ID. The need to change a given requirement is predicted using past requirement changes. In this work, all previously changed requirements are considered change initiators, and all requirements downstream from an initiator are considered change recipients. Though initial changes do not always propagate, industry members have confirmed the ECNs presented in this work to be the result of change propagation. This was verified by an engineer at each of the dataset's respective engineering sites. Table 10 displays the analyzed Project 3 ECNs. Ranking refers to a recipient's position in a list of requirements that is sorted according to similarity with the initiator, and depth reflects the percentage of the sorted list that must be read before arriving at the recipient. For example, computing depth among all requirements involves dividing a ranking by Project 2's total number of requirements.

The change prediction results obtained with the FC [CLS] embeddings are displayed in Table 11. The goal is to utilize requirement embeddings for identifying engineering changes, as these embeddings condense knowledge and information from

**Table 10 Project 3 approved ECNs**

| ECN | Requirements affected |
|---|---|
| 01 | R2.5.8-R2.1.2-R2.9.2-R2.1.14 |
| 07 | R2.1.14-R2.2.6 |
| 11 | R2.7 |

**Table 11 Project 3 change prediction results with FC [CLS] embeddings**

| | | Among All Requirements | | Among NFRs | |
|---|---|---|---|---|---|
| Recipient | Initiator | Ranking | Depth | Ranking | Depth |
| R2.2.6 | R2.5.8 | 56 | 26% | 53 | 31% |
| | R2.1.2 | 76 | 36% | 76 | 44% |
| | R2.9.2 | 137 | 63% | 131 | 76% |
| | R2.1.14 | 79 | 37% | 76 | 44% |
| R2.7 | R2.5.8 | 134 | 63% | 129 | 75% |
| | R2.1.2 | 157 | 73% | 150 | 87% |
| | R2.9.2 | 105 | 49% | 99 | 58% |
| | R2.1.14 | 151 | 71% | 144 | 84% |
| | R2.2.6 | 156 | 73% | 151 | 88% |

requirement documents. By utilizing the compressed information within the generated [CLS] token, this study explores alternative methods for identifying change propagations in requirement documents. The change prediction is based on the Faiss distance, which measures the similarity between the initial and subsequent requirements in vector space. By projecting them into a lower-dimensional space and utilizing the Faiss algorithm for similarity search and vector clustering based on distance, it becomes possible to estimate the propagation of requirement changes. Note that R2.1.14 is ignored as a recipient in ECN07 since it is also one of the change initiators affected by ECN01. Of the five initiators in ECN01, R2.5.8 is found to be the best predictor of downstream change in the recipient, R2.2.6, while the other analyzed recipient, R2.7, is best predicted by R2.9.2. For each result, there are improvements in ranking when analyzed among NFRs, but search depth among NFRs increases compared to search depth among all requirements. Maintaining a nearly equivalent ranking in a shorter list produces the observed increase in search depth.

*Comparison to SB Results.* The change prediction results generated with the FC [CLS] embeddings are compared to those produced by the SB embeddings, shown in Table 12. Each recipient is best predicted by a different initiator from those shown in Table 11. With a nearly perfect ranking and search depth of 1% and 2% among all requirements and NFRs, respectively, recipient R2.2.6 is best predicted by R2.1.14. Predicting recipient R2.7 proves to be a greater challenge, with R2.5.8 yielding the best depths of 31% among all requirements and 30% among NFRs.

The SB embedding results exhibit the opposite pattern of the FC [CLS] embedding results, as the search depth among NFRs is frequently less than the search depth among all requirements. Removing FRs from the list of considered requirements can be interpreted as eliminating "noise" from the change prediction search. The improvement in search depth signifies that when considering all requirements, enough FRs were ranked ahead of the recipient to obscure its ranking with respect to other NFRs. That is, removing FRs from Project 3's change prediction search improves the relevance of results, albeit by a small amount. Designers may benefit from separating FRs and NFRs before conducting a change prediction search to capture the most relevant results. If it is evident that change could only propagate to either FRs or NFRs, as is the case in Project 3, then only those requirements should be included in the change prediction search.

**Table 12 Project 3 change prediction results with SB embeddings**

| | | Among all requirements | | Among NFRs | |
|---|---|---|---|---|---|
| Recipient | Initiator | Ranking | Depth | Ranking | Depth |
| R2.2.6 | R2.5.8 | 79 | 37% | 62 | 36% |
| | R2.1.2 | 13 | 6% | 12 | 7% |
| | R2.9.2 | 72 | 34% | 47 | 27% |
| | R2.1.14 | 3 | 1% | 3 | 2% |
| R2.7 | R2.5.8 | 66 | 31% | 52 | 30% |
| | R2.1.2 | 128 | 60% | 99 | 58% |
| | R2.9.2 | 90 | 42% | 62 | 36% |
| | R2.1.14 | 86 | 40% | 64 | 37% |
| | R2.2.6 | 97 | 45% | 76 | 44% |

Overall, the SB embeddings clearly outperform the FC [CLS] embeddings when applied to change prediction. However, the noted improvement in search depth among NFRs suggests that a combination of the two models may be ideal: first, requirements could automatically be classified as FRs or NFRs with the FC BERT model, and then, SB embeddings could predict change in either set of requirements to yield the most relevant results.

## Discussion

Having completed each step in the research map presented in Fig. 1, the acquired results and observations are used to answer this work's three research questions.

**RQ1:** How may fine-tuned BERT's embedding of requirements be used to classify requirements based on source project?

Based on the MCC of 0.95 achieved on the PDC task, this work demonstrates that BERT can indeed differentiate across requirements documents. By obtaining a nearly perfect MCC, BERT proves its ability to identify the nuances that distinguish and relate requirements documents. Even though its pre-training corpus does not contain requirements, transfer learning allows BERT to be fine-tuned to recognize the semantic and syntactic patterns specific to requirements documents.

**RQ2:** How can the fine-tuned BERT embeddings of requirements be effectively utilized for accurate classification of FRs and NFRs in individual projects?

With an MCC of 0.82 computed for the FC task, this work indicates that BERT can also differentiate between FRs and NFRs, further demonstrating the level of detail contained in BERT's requirement representations. Not only can BERT recognize inter-document requirement patterns, but also intra-document requirement patterns. The successful classification results at these two levels of granularity suggest that BERT is suitable for a broad range of requirements management applications.

**RQ3:** How does a fine-tuned BERT model's capability to capture requirement change implications compare to general-language embeddings when adjusting requirements in existing systems?

This research question does not have a clear yes or no answer; the relative performance of BERT's fine-tuned embeddings depends on both the fine-tuning task and the particular requirements management application. Therefore, this question is addressed by considering how the PDC and FC models' [CLS] embeddings compare to the SB embeddings in each of the presented applications.

When computing similarity at the document level, the PDC [CLS] embeddings appear to have an advantage over the SB embeddings. The PDC [CLS] embeddings represent each document distinctly and exhibit clearly recognizable patterns between them, only some of which are reflected in the SB embeddings. Conversely, the SB embeddings are better equipped to perform requirement similarity searches, as they represent individual requirements distinctly and search across the entire dataset, whereas the PDC [CLS] embeddings tend to search only among the document closest to the query requirement's document.

In every explored instance of change prediction, the SB embeddings yielded better predictions than the FC [CLS] embeddings. As noted in the results section, however, in some cases the most relevant results are produced by using the two models in tandem: the FC BERT model groups requirements into FRs and NFRs, and then a

similarity search is performed within a group via SB embeddings. This approach relies on designers to anticipate which group, either FRs or NFRs, is most likely to be affected by an initial change.

In summary, the embedding performance depends on how well the fine-tuning task relates to the requirements management application. For instance, the SB embeddings are generated by a model that is fine-tuned specifically to represent general-language sequences for similarity searches, while the PDC and FC [CLS] embeddings are extracted from models that are fined-tuned only to recognize certain types of similarity, i.e., similarity based on parent document and similarity based on functionality. Consequently, the SB embeddings yield superior performance in both applications that involve a similarity search among requirements. The SB embeddings fail to outperform the PDC [CLS] embeddings when computing document similarities, however, because the PDC task specifically requires the BERT model to learn embeddings that distinguish documents. This work's recommendation is to use general-language sequence embeddings, such as those produced by the SB model, in requirements management applications unless sufficient labeled data exists to fine-tune a model for the exact desired application.

**Impact on Design Research and Practice.** With respect to design research, this work reaffirms the findings of previous studies that have applied transformer-based models to analyze design requirements. Having verified such models' ability to create detailed requirement embeddings, this work motivates future studies to further explore fine-tuning requirements management tasks. Currently, one of the greatest challenges for researchers in computational design is the lack of publicly available, labeled requirements datasets. Our contribution goes beyond the mere application of BERT to the management of engineering requirements. Using a novel dataset, we refine BERT specifically for requirement classification tasks, providing valuable insights into its performance. Our research uniquely emphasizes the fine-tuning of BERT for requirement data, and its effectiveness in discriminating design documents—an area inadequately covered in current literature. This paper explores a new domain, primarily aiming to understand classification mechanisms. As the framework matures to a stage ready for formal evaluation, it will be benchmarked against diverse established models. Aside from intellectual contributions, this work contributes to the design research community by releasing a portion of its labeled dataset. Though the remaining documents used in this work cannot be released, Documents 4 and 5 (580 requirements combined) labeled for the FC task are available on GitHub.[4]

This work's potential impact on design practice is envisioned through a theoretical scenario. Recall the period in the COVID-19 pandemic when there was a shortage of ventilators. Several companies with little relevant experience expressed a desire to begin producing ventilators. Such a company would benefit from determining which of its previously completed projects have the greatest overlap with ventilator design. After analyzing the design documentation, a transformer-based model could promptly rank projects according to similarity, allowing the company to get a quick start based on existing work. Once the company begins designing, their inexperience manifests in the need for many ECNs throughout the design process. Rather than repeatedly scanning through an entire requirements document for potential change propagation paths, a designer could use a transformer-based model to generate a sorted list of most likely change recipients, allowing more time to make the necessary preparations for any potential future change. Though idealized, this scenario provides a glimpse of how automated requirements management tools built on transformer-based models could empower designers to maximize their agility while minimizing unnecessary costs and delays.

[4]https://github.com/jessemullis/SKA_docs_ForNF

**Limitations.** Several limitations must be addressed to put this work's contributions into context. First, this work's dataset is not verified to be representative of all system requirements. Additionally, the dataset is large when compared to available collections of system requirements but remains dwarfed by other fine-tuning datasets. For instance, the SB model is fined-tuned on 215 M question–answer pairs from various sources. Increasing this work's dataset by at least an order of magnitude could improve generalizability. The PDC and FC tasks also vary in their potential to create generalizable models. A model trained on the PDC task is only applicable to documents included in its training dataset; the model's task is to classify requirements according to the documents it was trained on and those documents only. Since all requirements can be classified as FRs or NFRs, models trained on the FC task can generalize, but only if definitions of FRs and NFRs remain consistent. Finally, it may prove difficult to exactly replicate this work's results, even with the same dataset. Prior to fine-tuning, some of BERT's parameters are randomly initialized, and there is no guarantee that models will approach the same optimum. Several models were fine-tuned throughout this work, and each yielded slightly different results, though the general patterns remained consistent. It is important to note that some of the models may require training and retraining as they are used in industry. We do not anticipate significant retraining of FR and NFR as those are system-neutral. However, as new system—particularly novel ones—emerges in a company's portfolio, retraining/fine-tuning may be necessary to ensure the new classification is formally recognized. While this process is tedious, it can be facilitated using Kappa statistics to assist in evaluating the consistency of labeling, thus improving the performance and adaptability of the models.

Beyond BERT, there are several other methods in current literature to compute document embeddings. Word2Vec and GloVe are popular choices that utilize different mechanisms, such as context windows or co-occurrence statistics, to generate vector representations of words. Doc2Vec extends Word2Vec's principles to entire documents. Transformer-based models, like GPT and its successors, have also been effectively used for document embeddings. The scope of this study, however, is limited to the exploration of BERT models.

There are several limitations associated with the use of [CLS] embeddings. First, the incorporation of a pretrained model could potentially introduce biases into downstream applications. Second, task-specific models, like PDC and FC, may not consistently yield optimal results. Third, compared to other models such as TF-IDF, [CLS] tokens are more challenging to interpret and fail to capture sequential or contextual relationships adequately. Fourth, [CLS] tokens are typically capped at a length of 768, which could limit their effectiveness. Fifth, the average treatment of [CLS] embeddings, representing the entire sentence, could lead to a loss of nuanced information within texts. Finally, future studies should examine the differences in performance among the embeddings generated by trained [CLS] and fine-tuned SB models.

BERT-based models are notably effective for comprehending the overall semantics and the contextual relationships between requirements. When it comes to numerical data, these models may yield limited interpretability or precision. Although our approach may not precisely quantify specific numerical variations, it delivers a valuable understanding of broader patterns and changes in requirement content. To the best of our knowledge, no specific studies have addressed this challenge of capturing numerical value variations in the context of our research. Nonetheless, we intend to investigate the influence of such numerical value changes on change propagation in future work [13,44,61].

## Conclusion and Future Work

The immense potential for NLP in requirements management applications has long been established. Yet, no previous work has determined whether state-of-the-art NLP models, like BERT, can

sufficiently represent and distinguish requirements. Using a diverse set of 1303 requirements sourced from five system design projects, this work confirms that BERT can differentiate between requirements according to both parent document and functionality. After fine-tuning BERT on the PDC and FC tasks, the models' requirement-specific [CLS] embeddings are compared to general-language SB embeddings in requirements management applications. Namely, this work applies requirement embeddings to compute similarity and predict change. The PDC [CLS] embeddings outperform the SB embeddings for identifying document-level similarity, but the SB embeddings are superior for requirement-level similarity searches and change prediction.

This work supports the development of automated requirements management tools that rely on NLP models. With NLP's rapid progression in recent years, language embeddings have become increasingly capable of automatically revealing relationships between documents and requirements. These relationships can be unexpected and may guide designers to uncover patterns that would be missed otherwise. Though similarity and change prediction searches may not always produce relevant results, automated requirements management tools are not meant to replace humans entirely; rather, these tools' purpose is to provide suggestions that can be pursued or ignored at a designer's discretion.

Two branches of future work are suggested: one that investigates requirement embeddings and one that explores their applications in requirements management. To further examine requirement embeddings, future work could compare embeddings from each of BERT's layers, as it remains unclear which is most informative for downstream applications. Additionally, BERT is only one of many transformer-based models, so future work may involve comparing BERT's requirement embeddings with those generated by alternative models, such as MPNet [62]. To further investigate requirements management applications, a study that gathers human evaluations of requirement similarity within the dataset would provide a baseline for judging a model's requirement similarity analysis. While this work compared whole documents by averaging their requirements' embeddings, other methods of representing documents may be preferable. Finally, the SB embeddings yielded promising change prediction results that future work could build upon by considering the joint effect of multiple change initiators.

## Conflict of Interest

There are no conflicts of interest.

## Data Availability Statement

The datasets generated and supporting the findings of this article are obtainable from the corresponding author upon reasonable request.

## References


[1] Lops, P., De Gemmis, M., and Semeraro, G., 2011, "Content-Based Recommender Systems: State of the Art and Trends," *Recommender Systems Handbook*, Springer, New York, pp. 73–105.

[2] Alhindawi, N. T., 2018, "Information Retrieval-Based Solution for Software Requirements Classification and Mapping," Proceedings of the 2018 5th International Conference on Mathematics and Computers in Sciences and Industry (MCSI), Confu, Greece, Aug. 25–27, IEEE, pp. 147–154.

[3] Yadla, S., Hayes, J. H., and Dekhtyar, A., 2005, "Tracing Requirements to Defect Reports: An Application of Information Retrieval Techniques," Innov. Syst. Softw. Eng., **1**(2), pp. 116–124.

[4] Hu, H., Wen, Y., Chua, T.-S., and Li, X., 2014, "Toward Scalable Systems for Big Data Analytics: A Technology Tutorial," IEEE Access, **2**, pp. 652–687.

[5] Rajpathak, D., Peranandam, P. M., and Ramesh, S., 2022, "Automatic Development of Requirement Linking Matrix Based on Semantic Similarity for Robust Software Development," J. Syst. Softw., **186**, p. 111211.

[6] Mihany, F. A., Moussa, H., Kamel, A., Ezzat, E., and Ilyas, M., 2016, "An Automated System for Measuring Similarity Between Software Requirements," Proceedings of the 2nd Africa and Middle East Conference on Software Engineering, Cairo, Egypt, May 28–29, pp. 46–51.

[7] Zamani, K., 2021, "A Prediction Model for Software Requirements Change Impact," 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, Australia, Nov. 15–19, IEEE, pp. 1028–1032.

[8] Pahl, G., and Beitz, W., 2015, *Engineering Design: A Systematic Approach*, Springer Science+ Business Media, Berlin.

[9] Krueger, C., 2001, "Easing the Transition to Software Mass Customization," International Workshop on Software Product-Family Engineering, Springer, Berlin Heidelberg, pp. 282–293.

[10] Hein, P. H., Kames, E., Chen, C., and Morkos, B., 2021, "Employing Machine Learning Techniques to Assess Requirement Change Volatility," Res. Eng. Des., **32**(2), pp. 245–269.

[11] Hein, P. H., Kames, E., Chen, C., and Morkos, B., 2022, "Reasoning Support for Predicting Requirement Change Volatility Using Complex Network Metrics," J. Eng. Des., **33**(11), pp. 811–837.

[12] Ahmad, N., Wynn, D. C., and Clarkson, P. J., 2012, "Change Impact on a Product and Its Redesign Process: A Tool for Knowledge Capture and Reuse," Res. Eng. Des., **24**(3), pp. 219–244.

[13] Morkos, B., Mathieson, J., and Summers, J. D., 2014, "Comparative Analysis of Requirements Change Prediction Models: Manual, Linguistic, and Neural Network," Res. Eng. Des., **25**(2), pp. 139–156.

[14] de Araújo, A. F., and Marcacini, R. M., 2021, "Re-Bert: Automatic Extraction of Software Requirements From App Reviews Using Bert Language Model," Proceedings of the 36th Annual ACM Symposium on Applied Computing, Virtual, May 22–26, New York, pp. 1321–1327.

[15] Li, Y., and Yang, T., 2018, "Word Embedding for Understanding Natural Language: A Survey," *Guide to Big Data Applications*, Vol. 26, Springer International Publishing, Cham, Switzerland, pp. 83–104.

[16] Hey, T., Keim, J., Koziolek, A., and Tichy, W. F., 2020, "Norbert: Transfer Learning for Requirements Classification," Proceedings of the 2020 IEEE 28th International Requirements Engineering Conference (RE), Zurich, Switzerland, Aug. 31–Sept. 4, IEEE, pp. 169–179.

[17] Akay, H., and Kim, S.-G., 2020, "Measuring Functional Independence in Design With Deep-Learning Language Representation Models," Procedia CIRP, **91**, pp. 528–533.

[18] Lin, J., Liu, Y., Zeng, Q., Jiang, M., and Cleland-Huang, J., 2021, "Traceability Transformed: Generating More Accurate Links with Pre-Trained Bert Models," Proceedings of the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid Spain, May 25–28, IEEE, pp. 324–335.

[19] Abbas, M., Ferrari, A., Shatnawi, A., Enoiu, E., Saadatmand, M., and Sundmark, D., 2022, "On the Relationship Between Similar Requirements and Similar Software," Requir Eng., **28**(1), pp. 23–27.

[20] Navarro-Almanza, R., Juarez-Ramirez, R., and Licea, G., 2017, "Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification," Proceedings of the 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), Mérida, Mexico, Oct. 25–27, IEEE, pp. 116–120.

[21] Chen, C., Wei, S., and Morkos, B., 2023, "Bridging the Knowledge Gap Between Design Requirements and CAD-A Joint Embedding Approach," Proceedings of the 2023 ASEE Annual Conference & Exposition, Baltimore, MD, June 25–28.

[22] Karl, T., and Ulrich, S. E., 2018, *Product Design and Development*, 5th ed., McGraw-Hill Higher Education, Boston, MA.

[23] Kamalrudin, M., Mustafa, N., and Sidek, S., 2018, "A Template for Writing Security Requirements," Proceedings of the Requirements Engineering for Internet of Things: 4th Asia-Pacific Symposium, APRES 2017, Melaka, Malaysia, Nov. 9–10, Proceedings 4, Springer, pp. 73–86.

[24] Condamines, A., and Warnier, M., 2016, "Towards the Creation of a CNL Adapted to Requirements Writing by Combining Writing Recommendations and Spontaneous Regularities: Example in a Space Project," Lang. Resour. Eval., **51**(1), pp. 221–247.

[25] 2019, "Guide for Writing Requirements," https://www.incose.org/docs/default-source/working-groups/requirements-wg/rwg_products/incose_rwg_gtwr_summary_sheet_2022.pdf?sfvrsn=a95a6fc7_2, Accessed June 13, 2023.

[26] 2021, Simplified Technical English, AeroSpace and Defense Industries, Association of Europe, Issue 8. https://technicalwritingexpert.com/wp-content/uploads/2021/11/ASD-STE100-ISSUE-8.pdf

[27] Summers, J. D., Joshi, S., and Morkos, B., 2014, "Requirements Evolution: Relating Functional and Non-Functional Requirement Change on Student Project Success," Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Buffalo, NY, Aug. 17–20, American Society of Mechanical Engineers, p. V003T04A002.

[28] Shankar, P., Morkos, B., Yadav, D., and Summers, J. D., 2020, "Towards the Formalization of Non-Functional Requirements in Conceptual Design," Res. Eng. Des., **31**(4), pp. 449–469.

[29] Chung, L., and do Prado Leite, J. C. S., 2009, "On Non-Functional Requirements in Software Engineering,"*Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos*, Springer, Berlin/Heidelberg, pp. 363–379.

[30] Kurtanović, Z., and Maalej, W., 2017, "Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning," Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, Sept. 4–8, IEEE, pp. 490–495.

[31] Boetticher, G., 2007, "The PROMISE Repository of Empirical Software Engineering Data," http://promisedata.org/repository

[32] Winkler, J., and Vogelsang, A., 2016, "Automatic Classification of Requirements Based on Convolutional Neural Networks," Proceedings of the 2016 IEEE 24th International Requirements Engineering Conference Workshops (REW), Beijing, China, Sept. 12–16, IEEE, pp. 39–45.
[33] Akay, H., and Kim, S.-G., 2020, "Design Transcription: Deep Learning Based Design Feature Representation," CIRP Ann., **69**(1), pp. 141–144.
[34] Hayes, J. H., Dekhtyar, A., and Sundaram, S. K., 2006, "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods," IEEE Trans. Software Eng., **32**(1), pp. 4–19.
[35] Eder, S., Femmer, H., Hauptmann, B., and Junker, M., 2015, "Configuring Latent Semantic Indexing for Requirements Tracing," Proceedings of the 2015 IEEE/ACM 2nd International Workshop on Requirements Engineering and Testing, Florence, Italy, May 18, IEEE, pp. 27–33.
[36] Guo, J., Cheng, J., and Cleland-Huang, J., 2017, "Semantically Enhanced Software Traceability Using Deep Learning Techniques," Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, May 20–28 IEEE, pp. 3–14.
[37] Ferrari, A., Spagnolo, G. O., and Gnesi, S., 2017, "Pure: A Dataset of Public Requirements Documents," Proceedings of the 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, Sept. 4–8, IEEE, pp. 502–505.
[38] Clarkson, J., and Eckert, C., 2010, *Design Process Improvement: A Review of Current Practice*, Springer-Verlag , London.
[39] Eckert, C., Clarkson, P. J., and Zanker, W., 2004, "Change and Customisation in Complex Engineering Domains," Res. Eng. Des., **15**(1), pp. 1–21.
[40] Morkos, B., Shankar, P., and Summers, J. D., 2012, "Predicting Requirement Change Propagation, Using Higher Order Design Structure Matrices: An Industry Case Study," J. Eng. Des., **23**(12), pp. 905–926.
[41] Hein, P. H., Menon, V., and Morkos, B., 2015, "Exploring Requirement Change Propagation Through the Physical and Functional Domain," Proceedings of the ASME Design Engineering Technical Conference, Boston, MA, Aug. 2–5, New York, pp. 1–12.
[42] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., 2019, "Bert: Pre-Training of Deep Bidirectional Transformers for Language Understanding," Proceedings of NAACL-HLT , Minneapolis, MN, June 2–7.
[43] Taylor, W. L., 1953, "'Cloze Procedure': A New Tool for Measuring Readability," J. Q., **30**(4), pp. 415–433.
[44] Hein, P. H., Voris, N., and Morkos, B., 2017, "Predicting Requirement Change Propagation Through Investigation of Physical and Functional Domains," Res. Eng. Des., **29**(2), pp. 309–328.
[45] Hein, P. H., Morkos, B., and Sen, C., 2017, "Utilizing Node Interference Method and Complex Network Centrality Metrics to Explore Requirement Change Propagation," Proceedings of the Volume 1: 37th Computers and Information in Engineering Conference, Cleveland, OH, p. V001T02A081.
[46] Chen, C., Mullis, J., and Morkos, B., 2021, "A Topic Modeling Approach to Study Design Requirements," Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Online, Virtual.
[47] Cheng, C., and Morkos, B., 2023, "Exploring Topic Modelling for Generalising Design Requirements in Complex Design," J. Eng. Des., pp. 1–19.
[48] Reimers, N., and Gurevych, I., 2019, "Sentence-Bert: Sentence Embeddings Using Siamese Bert-Networks," Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China, Nov. 3–7.
[49] Lin, Y., Tan, Y. C., and Frank, R., 2019, "Open Sesame: Getting Inside BERT's Linguistic Knowledge," Association for Computational Linguistics, Florence, Italy, July 28–Aug. 2.
[50] Mullis, J., 2022, "Efficacy of Deep Neural Networks in Natural Language Processing for Classifying Requirements by Origin and Functionality: An Application of Bert in System Requirements," Dissertation, University of Georgia, Athens, GA.
[51] Cohen, J., 1960, "A Coefficient of Agreement for Nominal Scales," Educ. Psychol. Meas., **20**(1), pp. 37–46.
[52] McHugh, M. L., 2012, "Interrater Reliability: The Kappa Statistic," Biochem. Med. (Zagreb), **22**(3), pp. 276–282.
[53] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., and Funtowicz, M., 2020, "Transformers: State-of-the-Art Natural Language Processing," Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Online.
[54] Jurman, G., Riccadonna, S., and Furlanello, C., 2012, "A Comparison of MCC and CEN Error Measures in Multi-Class Prediction," PLOS One, **7**(8), pp. 1–9.
[55] Chicco, D., Tötsch, N., and Jurman, G., 2021, "The Matthews Correlation Coefficient (MCC) Is More Reliable Than Balanced Accuracy, Bookmaker Informedness, and Markedness in Two-Class Confusion Matrix Evaluation," BioData Min., **14**(1), pp. 1–22.
[56] Van der Maaten, L., and Hinton, G., 2008, "Visualizing Data Using T-SNE," J. Mach. Learn. Res., **9**(11), pp. 2579–2605.
[57] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., and Dubourg, V., 2011, "Scikit-Learn: Machine Learning in Python," J. Mach. Learn. Res., **12**, pp. 2825–2830.
[58] Aggarwal, C. C., Hinneburg, A., and Keim, D. A., 2001, "On the Surprising Behavior of Distance Metrics in High Dimensional Space," Proceedings of the Database Theory—ICDT 2001: 8th International Conference, London, UK, Jan. 4–6, Springer, pp. 420–434.
[59] Johnson, J., Douze, M., and Jegou, H., 2021, "Billion-Scale Similarity Search With GPUs," IEEE Trans. Big Data, **7**(3), pp. 535–547.
[60] Hinkle, D. E., Wiersma, W., and Jurs, S. G., 2003, *Applied Statistics for the Behavioral Sciences*, Vol. 663, Houghton Mifflin College Division, Boston, MA.
[61] Shankar, P., Morkos, B., and Summers, J. D., 2012, "Reasons for Change Propagation: A Case Study in an Automotive OEM," Res. Eng. Des., **23**(4), pp. 291–303.
[62] Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y., 2020, "Mpnet: Masked and Permuted Pre-Training for Language Understanding," Adv. Neural Inf. Process. Syst., **33**, pp. 16857–16867.