Transfer learning for conflict and duplicate detection in software requirement pairs

1st Garima Malik
Toronto Metropolitan University
Toronto, Canada
garima.malik@torontomu.ca

2nd Savas Yildirim
Toronto Metropolitan University
Toronto, Canada
savas.yildirim@torontomu.ca

3nd Mucahit Cevik
Toronto Metropolitan University
Toronto, Canada
mcevik@torontomu.ca

arXiv:2301.03709v3 [cs.SE] 25 Jul 2026

***Abstract*—Conflicting and duplicate software requirements can lead to design errors, increased development cost, and project delays if they are not identified early. However, automatically detecting such inconsistencies is difficult because software requirements are written in natural language and often contain complex semantics. In this work, we formulate conflict and duplicate detection as a requirement pair classification task and propose *SR-BERT*, a transformer-based framework that combines bi-encoder architectures with domain adaptation. To evaluate the proposed approach, we construct two proprietary datasets, namely CDN and CN, with the help of software development experts and transform four publicly available requirement datasets into requirement pair classification datasets. We then investigate sequential multi-stage fine-tuning and cross-domain transfer learning strategies across these datasets. Experimental results show that sequential fine-tuning consistently improves performance over baseline models, with SR-BERT achieving the best results on larger datasets. Cross-domain experiments further demonstrate that the proposed framework generalizes well across both industrial and open-source requirements. Overall, our findings show that domain-adapted transformer models provide an effective solution for automating software requirement conflict and duplicate detection.**

***Index Terms*—Software requirements, conflict and duplicate detection, transformer models, transfer learning, cross-domain**

I. Introduction

Requirement Engineering (RE) is a fundamental phase of software development that focuses on identifying, documenting, and managing software requirements. These requirements are usually captured in a Software Requirements Specification (SRS) document, which describes the expected system functionalities and constraints. To ensure successful software development, the requirements should be accurate, consistent, and complete. Poor-quality requirements can lead to design errors, increased development costs, project delays, and reduced user satisfaction [1, 2, 2]. However, SRS documents often contain inconsistencies such as conflicting requirements, where one requirement contradicts another, and duplicate requirements, where the same requirement is expressed using different wording. If these issues are not detected early, they can negatively affect software quality and system reliability [3, 4, 5].

Automatically detecting conflicting and duplicate requirements remains challenging for several reasons. First, conflicting requirement pairs are relatively rare, resulting in severe class imbalance during model training [4]. Second, software requirements often contain domain-specific terminology and depend on project context, making it difficult to determine whether two requirements truly conflict [3]. Third, the same requirement can be written in many different ways, with variations in wording and sentence structure, making duplicate detection challenging [6]. Finally, software requirements evolve throughout the development process, and requirements that are initially consistent may become conflicting as the system changes [7].

Recent advances in Natural Language Processing (NLP), especially pretrained transformer models such as BERT [8] and RoBERTa [9], have significantly improved performance on a wide range of sentence classification tasks through transfer learning. In particular, the Natural Language Inference (NLI) task, which determines whether one sentence entails, contradicts, or is neutral with respect to another, closely resembles the problem of analyzing requirement pairs. Motivated by this similarity, we formulate conflict and duplicate detection as a requirement pair classification task and further investigate a binary conflict vs neutral formulation for conflict detection.

The main contributions of this work is as follows:

- We formulate conflict and duplicate identification as a requirement pair classification task and curate two expert-annotated datasets, CDN (Conflict, Duplicate, and Neutral) and CN, constructed following industry guidelines with requirements authored by software engineering professionals.
- We demonstrate the effectiveness of sequential transfer learning by fine-tuning BERT checkpoints first on large-scale NLI datasets and subsequently on requirement pair classification data, establishing a strong baseline for the task.
- We propose SR-BERT, a domain-adapted bi-encoder architecture that leverages pretrained Sentence-BERT and further adapts it to the software requirements domain through unsupervised domain adaptation, enabling more accurate semantic encoding of requirement pairs.
- We systematically evaluate cross-domain transfer learning to assess the generalizability of BERT-based checkpoints across diverse requirement datasets, providing insights into their capability in real-world deployment scenarios.

II. Literature Review

A. NLP and Transfer Learning in Requirements Engineering

Natural Language Processing (NLP) has become central to automating requirements engineering (RE) tasks, including requirement classification, quality assessment, and information extraction [10, 11]. Recent work has shifted from lexical and syntactic features toward deep embedding techniques, demonstrating strong performance across semantic-level RE tasks [12]. Transfer learning, particularly through pretrained transformer models such as BERT [8] and DeBERTa [13], has further advanced the field by reducing dependence on large labeled datasets while improving generalization [14]. In RE specifically, transfer learning has been successfully applied to requirement classification [15, 16], anaphoric ambiguity detection [17], and domain concept extraction [18]. Studies have shown that domain-adapted language models (e.g., NoRBERT, BERT4RE) outperform generic pretrained models on software requirement texts [15, 18]. However, industrial validation remains limited due to scarce public RE datasets [11].

B. Sentence Pair Classification

Sentence pair classification underpins many NLP tasks relevant to RE, including semantic textual similarity (STS) [19], paraphrase identification [20], and NLI [21]. Transformer-based models, fine-tuned on large NLI datasets such as Multi-Genre NLI (MNLI) [22], have proven particularly effective for determining semantic relationships between text pairs. The label structure and scale of MNLI (433k instances) make it especially suitable for transfer learning to domain-specific pair classification tasks, including requirement comparison.

C. Conflict and Duplicate Identification

Automated detection of conflicting requirements remains a challenging problem in Requirement Engineering (RE). Existing approaches can be broadly grouped into four categories. The first category uses rule-based techniques that combine Natural Language Processing (NLP) pipelines with manually designed heuristics to identify conflicts [3, 23, 24]. The second category applies fuzzy logic to detect and resolve requirement conflicts [25, 26]. The third category employs deep learning models, particularly BiLSTM-based architectures, to identify conflicts in non-functional requirements [27]. More recently, transformer-based methods have been proposed, combining semantic similarity with entity overlap analysis to improve conflict detection [4].

Compared with conflict detection, duplicate requirement detection has received much less attention. Although duplicate detection has been widely studied for bug reports using CNN and RNN-based models [28, 29], similar research on software requirements is limited. Existing studies mainly focus on empirical analyses, and there is still a lack of well-established benchmarks and robust learning-based approaches for identifying duplicate requirements [30, 31].

III. Methodology

A. Datasets

We evaluate our approach on six datasets spanning software engineering, healthcare, transportation, and hardware: two proprietary datasets (CDN and CN) and four open-source SRS datasets (UAV, WorldVista, PURE, and OPENCOSS). When requirements are written as long paragraphs, we segment them into individual sentences to obtain a consistent requirement-pair representation. We first describe the proprietary datasets and then summarize the open-source datasets used in the cross-domain experiments.

- *CDN*: Derived from IBM-DOORS requirements and user stories [32], comprising three labels: Conflict, Duplicate, and Neutral. A team of 10 IBM software developers created conflict requirements for 48 original requirements, while nine developers composed duplicates. Domain experts verified all pairs. To mitigate class imbalance, neutral pairs were limited to half the sum of conflict and duplicate pairs. Table I illustrates the pair structure. For example, requirement pair (1,2) is labeled as a conflict because the modifier "only" and the alternative command sources ("pilot controller" versus "remote viewing app") imply incompatible access rules. Duplicate pairs instead use synonyms or closely related phrases, whereas neutral pairs are drawn from distinct domains or SRS documents with different actors and actions. Table II reports the class distribution, and Figure 1e shows the cosine-similarity distributions. The overlap between conflict and duplicate pairs in this embedding space illustrates why the three-class task is nontrivial.
- *CN*: We also use a binary variant, CN, derived from CDN by removing duplicate pairs to enable cross-domain transfer experiments. Figure 1f shows that conflict pairs exhibit higher cosine similarity than neutral pairs, as conflicting requirements often share similar verbs and actors (Table I). This structure makes CN a useful source dataset for testing whether conflict-relevant semantic patterns transfer to other requirement corpora.

We obtained the UAV, WorldVista, PURE, and OPENCOSS datasets from Malik et al. [4], this paper utilized these datasets to evaluate the supervised semantic similarity-based conflict detection approach (S3CDA). These datasets initially contained limited conflict instances; to enrich them, Malik et al. [4] supplemented each with synthetic conflict requirements crafted following INCOSE guidelines to align with industry standards. Table II summarizes the class distribution, average token length, and vocabulary size for each dataset.

- *UAV*: Requirements for a UAV control system from the University of Notre Dame, structured using the EARS template [33]. Figure 1b shows clear separation in cosine similarity between neutral and conflict labels.
- *WorldVista*: Health management system requirements[2] with basic healthcare terminology. This dataset contains

[2]https://worldvista.org/Documentation

TABLE I: Example data instances from the CDN dataset, showing requirement pairs and their respective labels, with emphasis on words and phrases that indicate the assigned label.

Req. pair Id	Requirement Text 1	Requirement Text 2	Label
(1,2)	The UAV shall **only** accept commands from an authenticated **Pilot controller**.	The UAV shall accept commands from an authenticated **remote viewing app**.	Conflict
(3,4)	All UAV data transmission shall be secure against **unauthorized access**.	All UAV data transmission shall be secure against **all access**.	Conflict
(5,6)	These are the **client** requirements for the Aviary System of Systems.	These are the **customer** requirements for the Aviary System.	Duplicate
(7,8)	The UAV shall send the Pilot real-time information about **malfunctions** that **impact** the mission.	The UAV shall send the Pilot real-time data about **errors** that **affect** the mission.	Duplicate
(9,10)	The **Pilot** shall be able to **fly** the UAV to any accessible-by-air location within 20 miles of its origin	The **UAV** shall **charge** to 75% in less than 3 hours.	Neutral
(11,12)	The **UAV** shall periodically **send** the Pilot the power and estimated flight time remaining.	A **single adult** shall be able to **lift** and carry the UAV.	Neutral

TABLE II: Dataset statistics for requirement pair datasets.

Dataset	# Conflict	# Duplicate	# Neutral	Avg. # tokens in pairs	Vocabulary size
CDN	5,553	1,673	3,400	(18.19, 18.43)	1022
WorldVista	35	-	10,843	(23.09, 21.80)	884
UAV	18	-	6,652	(26.44, 26.30)	298
PURE	20	-	2,191	(26.44, 26.30)	298
OPENCOSS	10	-	6,776	(29.85, 30.02)	250
CN	5,553	-	3,400	(18.23, 18.51)	1022

35 conflict and 10,843 neutral pairs. Figure 1a shows similar distinct cosine similarity distributions across labels.

- *PURE*: Derived from two SRS documents (THEMAS and Mashbot) in the public PURE corpus [34]. Long paragraphs were segmented into individual sentences. The dataset contains 20 conflict and 2,191 neutral pairs. Figure 1c shows cosine similarity patterns comparable to the UAV dataset.
- *OPENCOSS*: Railway and automotive safety requirements[1]. With only 10 conflict pairs against 6,776 neutral pairs, Figure 1d exhibits a heavily skewed conflict label distribution.

B. Sequential Transfer Learning

Sequential transfer learning leverages knowledge from a source domain to improve target task performance by progressively adapting language model checkpoints [14]. While transfer learning is often described as pretraining followed by task-specific fine-tuning, recent work also considers multi-stage adaptation through intermediate labeled or unlabeled datasets, provided negative transfer is avoided [14]. Our experiments use both direct task fine-tuning and multi-stage supervised fine-tuning. Figure 2 illustrates the sequential pipeline: a vanilla BERT checkpoint is first fine-tuned on the MNLI dataset to learn general sentence-pair inference patterns and is then fine-tuned on requirement-pair datasets such as CDN to adapt those patterns to conflict and duplicate detection. To isolate the value of the intermediate NLI stage, we compare these sequential checkpoints with vanilla BERT variants that are fine-tuned directly on the requirement-pair datasets.

[1]http://www.opencoss-project.eu

C. SR-BERT

Sentence-BERT (SBERT) is an extension of BERT designed to produce semantically meaningful sentence embeddings while enabling efficient sentence-level comparisons [19]. Unlike the original BERT architecture, which jointly processes sentence pairs during inference, SBERT adopts a bi-encoder architecture that independently encodes each sentence into a fixed-length vector. This design significantly reduces the computational cost of pairwise comparison and makes SBERT well suited for tasks such as semantic similarity, information retrieval, and sentence pair classification.

SBERT is trained using siamese and triplet network architectures, in which multiple copies of the same encoder share parameters during training. The shared-weight design encourages the model to learn a representation space where semantically related sentences are placed closer together while unrelated sentences are pushed farther apart.

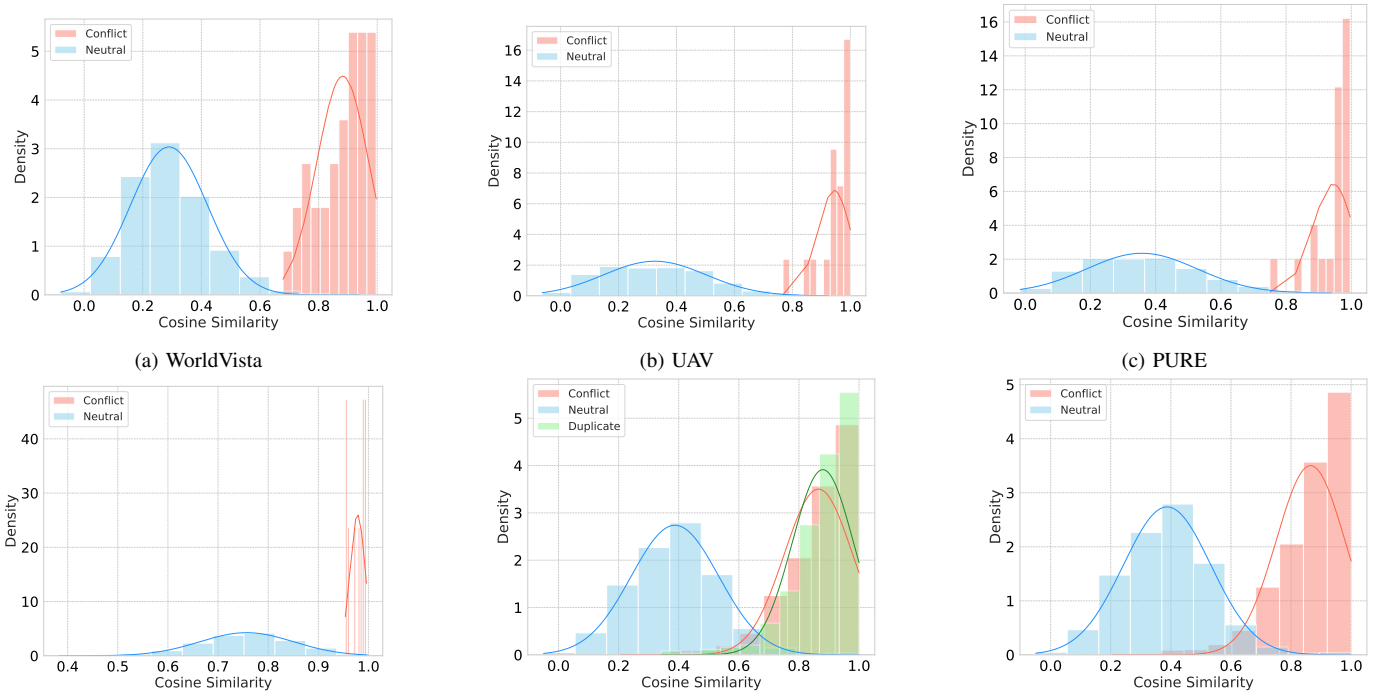
Figure 3 illustrates the use of SBERT for software requirement pair classification. Each requirement is independently encoded to obtain its sentence embedding, after which the embeddings are combined and passed to a linear classifier for prediction.

SBERT can be trained using either siamese or triplet learning objectives.

- *Siamese networks*: Two identical encoder networks independently process a pair of input sentences while sharing the same parameters. The objective is to minimize the embedding distance between semantically similar sentence pairs and maximize the distance between dissimilar pairs.
- *Triplet networks*: Three identical encoder networks process an anchor sentence, a semantically similar (positive) sentence, and a semantically dissimilar (negative) sentence. The model learns embeddings by simultaneously bringing the anchor closer to the positive example while pushing it away from the negative example.

The triplet learning objective is defined as:

$$L = \max\left(0,\, D(\mathbf{R}_A, \mathbf{R}_P) - D(\mathbf{R}_A, \mathbf{R}_N) + \epsilon\right), \qquad (1)$$



(a) WorldVista (b) UAV (c) PURE

(d) OPENCOSS (e) CDN (f) CN

Fig. 1: Cosine similarity distribution of requirement pairs for the respective class labels, computed using SBERT embeddings.

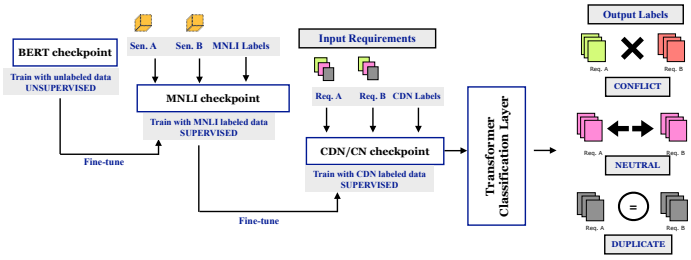


Fig. 2: A visual description of the sequential transfer learning approach for requirement pair classification

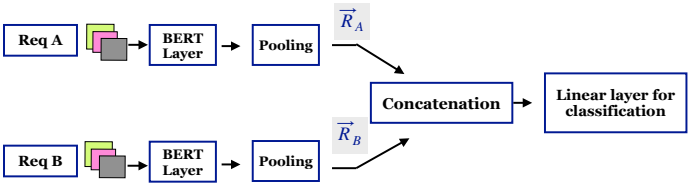


Fig. 3: SBERT-based model for software requirement pair classification. Individual requirements are encoded using SBERT to obtain sentence embeddings, which are subsequently concatenated and passed through a linear classifier.

where $\mathbf{R}_A$, $\mathbf{R}_P$, and $\mathbf{R}_N$ denote the embeddings of the anchor, positive, and negative requirements, respectively, $D(\cdot, \cdot)$ represents a distance metric (typically cosine or Euclidean distance), and ϵ is the margin that enforces a minimum separation between positive and negative pairs. For example,

- R_A: *The system shall allow users to reset their passwords.*
- R_P: *Users should be able to reset their passwords in the system.*
- R_N: *The software must provide real-time data synchronization.*

Here, R_A and R_P express the same functional requirement using different wording and therefore should be mapped to nearby locations in the embedding space. In contrast, R_N represents an unrelated functionality and should be placed farther away from the anchor representation.

A key advantage of SBERT is that sentence embeddings can be generated independently and reused across multiple downstream tasks without repeatedly encoding sentence pairs. Consequently, SBERT serves as an effective feature extractor for transfer learning, where the generated embeddings can be employed for semantic similarity, clustering, retrieval, or classification tasks. Compared with cross-encoder models, bi-encoder architectures such as SBERT offer lower pairwise inference cost while preserving sentence-level semantic representations.

Building upon these advantages, we propose *Software Requirement BERT (SR-BERT)*, a domain-adapted SBERT framework for software requirement pair classification. As illustrated in Figure 4, SR-BERT starts from a pretrained SBERT checkpoint and further fine-tunes it on labeled software requirement pairs using a supervised three-class classification objective. This domain adaptation enables the encoder to learn representations that better capture the semantic characteristics of software requirements.

Given a pair of requirements, the fine-tuned encoder generates two sentence embeddings, denoted by R_1 and R_2. Rather

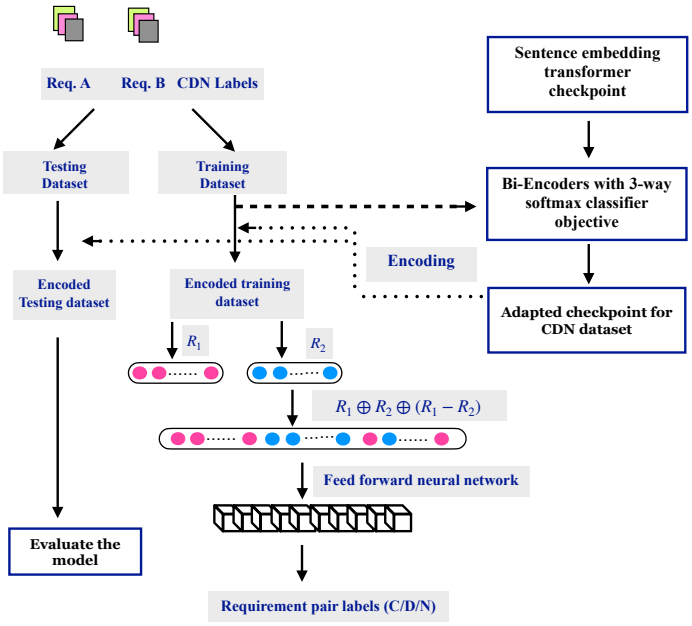


Fig. 4: Architecture of the proposed SR-BERT model.

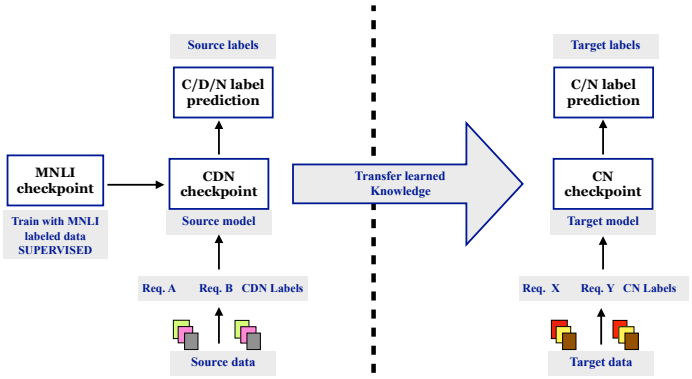


Fig. 5: Cross-domain transfer learning framework for requirement pair classification task.

than relying solely on their concatenation, we construct the final pair representation as

$$R = R_1 \oplus R_2 \oplus (R_1 - R_2), \tag{2}$$

where $\oplus$ denotes vector concatenation. The embeddings R_1 and R_2 preserve the semantic representation of each individual requirement, while the difference vector $(R_1 - R_2)$ explicitly captures their semantic relationship. Incorporating both individual and relational information enables the classifier to distinguish requirement pairs with subtle semantic differences more effectively than using either representation alone.

The resulting feature vector is then passed through a feed-forward neural network with a softmax output layer, which is trained using the ground-truth labels to perform requirement pair classification.

D. Baseline classification methods

To evaluate the effectiveness of SR-BERT, we compare it against three representative baseline approaches that differ only in the sentence representation used while employing a comparable linear classifier.

- *TF-IDF + SVM*: Each requirement is represented using TF-IDF features. The feature vectors corresponding to a requirement pair are concatenated and classified using a linear Support Vector Machine (SVM).
- *Word Embeddings (GloVe + FastText) + SVM*: Each requirement is represented by averaging pretrained GloVe and FastText word embeddings. The resulting sentence representations are concatenated and provided as input to a linear SVM classifier.
- *SBERT + Linear Layer*: Requirements are independently encoded using a pretrained SBERT model. The resulting sentence embeddings are concatenated and fed directly to a linear classification layer, as illustrated in Figure 3. Unlike SR-BERT, this baseline does not perform domain-specific adaptation of the SBERT encoder.

E. Cross-domain transfer learning

We use cross-domain transfer learning to test whether a transformer trained on one requirement-pair dataset can support classification in another dataset with the same label structure [14, 35, 36]. This experiment is motivated by shared requirement structure across the datasets and by the cosine-similarity patterns observed for conflict pairs. Formally, we define a source-domain dataset as $D_s = \left\{(r^i_{s_1}, r^i_{s_2}), y^i_{s_{1,2}}\right\}_{i=1}^{N_s}$ and a target domain dataset as $D_t = \left\{(r^i_{t_1}, r^i_{t_2}), y^i_{t_{1,2}}\right\}_{i=1}^{N_t}$, with (r_{s_1}, r_{s_2}) representing a requirement pair in the source dataset and $y_{s_{1,2}}$ representing the label defining the relationship between requirement pairs. Similarly, (r_{t_1}, r_{t_2}) represents a requirement pair in the target dataset, and $y_{t_{1,2}}$ represents the corresponding label.

Figure 5 illustrates the domain adaptation configuration for our experiments. In our requirement datasets, CDN and CN have relatively balanced class distributions, while the other requirement-pair datasets contain few conflict instances. We first train a transformer model on CN and evaluate it on the other requirement-pair datasets. We then train models on individual datasets and on combinations of UAV, WorldVista, OPENCOSS, and PURE, evaluating each model on the held-out dataset. This design allows us to assess whether shared structure in the input format and label definition is sufficient to support transfer across requirement sources.

F. Experimental setup

We evaluate the proposed methods using several pre-trained transformer models and their corresponding variants that have been further fine-tuned on the Multi-Genre Natural Language Inference (MNLI) dataset. Specifically, we consider three general-purpose pretrained language models, namely *distilbert-base*[3], *bert-base*[1], and *deberta-base*[5]. In addition, we evaluate three MNLI-adapted checkpoints, namely

[3]https://huggingface.co/distilbert-base-uncased
[1]https://huggingface.co/bert-base-uncased
[5]https://huggingface.co/microsoft/deberta-base

TABLE III: Transformer model checkpoint characteristics and hyperparameter details

Model checkpoints	Layers	Hidden units	Heads	Total params	Batch size	Epochs
bert-base-uncased	12	768	12	110M	32	5
deberta-base	12	768	12	100M	32	5
distilbert-base	6	768	12	66M	32	5
roberta-large-mnli	24	1024	16	355M	32	5
bert-base-uncased-MNLI	12	768	12	110M	32	10
deberta-base-mnli	12	768	12	86M	32	10

roberta-large-mnli[6], *bert-base-uncased-MNLI*[7], and *deberta-base-mnli*[8]. These models are further fine-tuned on the software requirement pair datasets before evaluation.

Table III summarizes the architectural characteristics of the transformer checkpoints together with the training hyperparameters adopted in our experiments.

For all transformer models, hyperparameter tuning is performed by exploring different combinations of learning rates, batch sizes, and training epochs. Batch sizes ranging from 8 to 128 and training epochs between 2 and 10 are evaluated using an adaptive learning-rate scheduler. Across all datasets, we observe that smaller batch sizes consistently provide better generalization than larger batches, while the optimal number of training epochs typically lies between five and ten. Early stopping based on the validation loss is employed to prevent overfitting and to retain the best-performing model checkpoint. All experiments are conducted on an NVIDIA P100 GPU equipped with 32 GB of memory.

The computational cost of fine-tuning largely depends on both the dataset size and the number of model parameters. Among the evaluated checkpoints, *roberta-large-mnli* requires approximately 1–2 hours of training, whereas the lighter models, namely *deberta-base-mnli* and *distilbert-base*, typically complete training within 10–15 minutes. The *bert-base-uncased-MNLI* checkpoint requires approximately 20–25 minutes to converge.

For the proposed SR-BERT framework, we initialize the sentence encoder using the pretrained *distilbert-base-nli-mean-tokens*[7] checkpoint, which is subsequently fine-tuned on the software requirement datasets to learn domain-specific sentence representations.

The datasets used in this study exhibit different class distributions. The CDN dataset contains three classes (Conflict, Duplicate, and Neutral) with relatively balanced class proportions, making it suitable for 5-fold cross-validation. In contrast, the remaining datasets consist of only Conflict and Neutral classes, with conflict pairs representing a small minority of the data. Consequently, 3-fold cross-validation is employed for these datasets. Table IV presents the class distributions for the training and testing partitions in each cross-validation fold.

For the proposed SR-BERT framework, each requirement pair is represented by three embedding vectors that are concatenated before being provided to the downstream classifier. Specifically, the representations correspond to the embeddings of the two individual requirements together with their element-wise difference. To provide a fair comparison, we also evaluate three baseline feature representations, namely TF-IDF, pooled GloVe–FastText embeddings, and SBERT embeddings. The corresponding feature extraction settings, SVM parameters, and SR-BERT hyperparameters are summarized in Table V.

[6]https://huggingface.co/roberta-large-mnli

[7]https://huggingface.co/textattack/bert-base-uncased-MNLI

[8]https://huggingface.co/microsoft/deberta-base-mnli

[7]https://huggingface.co/sentence-transformers/distilbert-base-nli-mean-tokens

TABLE IV: Class distribution in each fold's training and test sets for all the datasets.

		Training				Test			
Dataset	**# folds**	**# D**	**# C**	**# N**	**Total**	**# D**	**# C**	**# N**	**Total**
CDN	5	1,339	4,442	2,720	8,501	334	1,111	680	2,125
		Training				**Test**			
Dataset	**# folds**		**# C**	**# N**	**Total**		**# C**	**# N**	**Total**
CN	3		3,702	2,206	5,908		1,851	1,134	2,985
UAV	3		12	4,434	4,446		6	2,218	2,224
WorldVista	3		23	7,229	7,252		12	3,614	3,626
PURE	3		14	1,460	1,474		6	731	737
OPENCOSS	3		7	4,517	4,524		3	2,259	2,262

IV. Results

A. CDN dataset classification results

We compare SR-BERT, sequential fine-tuning checkpoints, and vanilla BERT against three baseline embeddings (TF-IDF, GloVe+FastText, and SBERT) on the CDN dataset using 5-fold cross-validation. Table VI reports weighted and macro precision, recall, and F1-scores. SR-BERT obtains the highest weighted F1-score (95.3%), followed by the sequentially fine-tuned MNLI checkpoints (*deberta-base-mnli*: 92.6%, *bert-base-uncased-MNLI*: 92.0%). The MNLI-adapted checkpoints improve over their vanilla counterparts, and the low standard deviations indicate stable performance across folds. Table VII presents class-specific performance for the best model. SR-BERT achieves high recall for conflict and neutral pairs, while duplicate pairs have lower precision, recall, and F1-score. This pattern is consistent with the smaller duplicate class and the semantic overlap between duplicate and conflict pairs.

B. Conflict and neutral classification results

We evaluate binary conflict-neutral classification using 3-fold cross-validation due to smaller dataset sizes and severe class imbalance. Table VIII reports baseline, MNLI fine-tuned, and SR-BERT performance, including conflict-specific metrics. On CN, the MNLI-adapted checkpoints and SR-BERT all achieve high conflict-class F1-scores, with *deberta-base-mnli* reaching the highest Conflict-F1 (99.6%) and *bert-base-uncased-MNLI* reaching the highest Macro-F1 (99.4%). On the open-source datasets, the baseline embeddings and SR-BERT mostly predict the majority neutral class, leading to Macro-F1 values near 49% and zero reported Conflict-F1. In contrast, the sequentially fine-tuned models retain conflict-detection ability under severe imbalance: *bert-base-uncased-MNLI* reaches Macro-F1 scores of 90.8% on WorldVista

TABLE V: Baseline embeddings with linear classifier (SVM) and SR-BERT hyperparameters.

	max_features	**ngram_range**	**Min_df**	**Vector length**	
TF-IDF	1500	(1,3)	5	1380	
	Embedding Size	**Context Window Size**	**Max Sentence Length**	**Embedding Type**	**Vector length**
FastText	4096	10	250	LSTM-based	500
GloVe	300	-	-	Static	300
	C	**Gamma**	**Kernel**	**Degree**	**Tolerance**
SVM	1.0	'auto'	Linear	3	0.001
	Loss function	**Epochs (max)**	**Learning rate (LR)**	**LR Scheduler**	**Batch size**
SBERT embeddings	Softmax and cosine similarity loss	3	1e-3	Exponential	16
SR-BERT embeddings	Softmax loss	5	1e-5	Exponential	16
	Dense units	**Epochs (max)**	**Activation function**	**Dropout**	**Batch size**
Feed-forward neural network	1500	50	ReLU	0.2	16

TABLE VI: Results for CDN dataset. Values are averaged over 5 folds and results are shown as "mean ± standard deviation".

	Weighted			**Macro**		
Models	Precision	Recall	F1	Precision	Recall	F1
TF-IDF + SVM	0.808 ± 0.013	0.807 ± 0.004	0.756 ± 0.006	0.804 ± 0.024	0.643 ± 0.005	0.629 ± 0.008
WE (Glove+FastText) + SVM	0.591 ± 0.006	0.530 ± 0.003	0.376 ± 0.007	0.505 ± 0.006	0.341 ± 0.003	0.246 ± 0.007
SBERT+linear layer	0.593 ± 0.018	0.622 ± 0.017	0.588 ± 0.016	0.586 ± 0.018	0.594 ± 0.035	0.571 ± 0.024
bert-base-uncased	0.797 ± 0.049	0.833 ± 0.007	0.779 ± 0.019	0.749 ± 0.106	0.675 ± 0.026	0.646 ± 0.045
distilbert-base	0.854 ± 0.006	0.866 ± 0.006	0.854 ± 0.007	0.813 ± 0.011	0.784 ± 0.012	0.794 ± 0.010
deberta-base	0.812 ± 0.048	0.848 ± 0.013	0.812 ± 0.033	0.758 ± 0.089	0.715 ± 0.056	0.706 ± 0.067
roberta-large-mnli	0.923 ± 0.011	0.916 ± 0.011	0.918 ± 0.011	0.883 ± 0.013	0.907 ± 0.017	0.892 ± 0.014
bert-base-uncased-MNLI	0.923 ± 0.011	0.919 ± 0.008	0.920 ± 0.009	0.890 ± 0.007	0.901 ± 0.020	0.894 ± 0.011
deberta-base-mnli	0.928 ± 0.003	0.924 ± 0.002	0.926 ± 0.002	0.895 ± 0.002	0.909 ± 0.006	0.901 ± 0.002
SR-BERT	**0.953 ± 0.005**	**0.953 ± 0.004**	**0.953 ± 0.005**	**0.936 ± 0.003**	**0.938 ± 0.010**	**0.937 ± 0.006**

TABLE VII: Class-specific performance with the best performing model, SR-BERT, over CDN dataset. Reported values are averaged over 5 folds and results are shown as "mean ± standard deviation".

	Precision	**Recall**	**F1**	**support**
Conflict	0.956 ± 0.011	0.956 ± 0.003	0.956 ± 0.004	1110.60
Duplicate	0.855 ± 0.015	0.871 ± 0.032	0.862 ± 0.012	334.60
Neutral	0.997 ± 0.002	0.989 ± 0.005	0.993 ± 0.002	680.00

and 84.1% on OPENCOSS, while *deberta-base-mnli* reaches 87.7% on UAV and 94.8% on PURE. These results indicate that NLI initialization is especially valuable for imbalanced binary conflict detection.

C. Cross-domain transfer learning results

We evaluate cross-domain transfer for binary conflict classification using 3-fold cross-validation with macro-averaged metrics (Table IX). Models trained on CN achieve high macro recall but low macro precision and very low Conflict-F1 on UAV, WorldVista, PURE, and OPENCOSS, indicating many false positives. This pattern suggests that CN's requirement structure does not transfer cleanly to the open-source datasets, so we exclude CN from the multi-source combinations. We find that the strongest source configuration depends on the target dataset. PURE transfers best to UAV (Macro-F1: 89.2%, Conflict-F1: 78.6%); the combined UAV+PURE source performs best on WorldVista (Macro-F1: 93.6%, Conflict-F1: 87.3%), exceeding the in-domain WorldVista Macro-F1 of 90.8%; and the combined UAV+WorldVista+OPENCOSS source performs best on PURE (Macro-F1: 89.8%). For OPENCOSS, UAV alone achieves the best cross-domain Macro-F1 (67.7%), suggesting that some source-target pairs share more transferable structure than broader multi-source combinations.

D. Comparison with Existing Techniques

We compare our approach against three baseline methods from the literature, ensuring fair evaluation by using identical datasets and folds where applicable.

a) Comparison with Malik et al. [4]: Table X reports results on the WorldVista, UAV, PURE, and OPENCOSS datasets. The sequentially fine-tuned models improve over their supervised semantic similarity approach (S3CDA) on all four datasets. The largest gains occur on WorldVista (*deberta-base-mnli*: 0.93 vs. 0.86) and OPENCOSS (*bert-base-uncased-MNLI*: 0.74 vs. 0.43), suggesting that transformer-based transfer learning is useful when requirements contain general-domain terminology or when the dataset is severely imbalanced (OPENCOSS: 10 conflict vs. 6,776 neutral pairs). On PURE and UAV, where cosine similarity already separates the classes more clearly, the margins are narrower but still favor the sequentially fine-tuned models.

b) Comparison with Abeba and Alemneh [27]: Table XI shows that their BiLSTM + Word2vec approach struggles with class imbalance, achieving near-random macro F1-scores (0.47–0.48) by predicting the majority class. In contrast, sequential fine-tuning of *bert-base-uncased-MNLI* yields higher macro F1-scores (0.79–0.86), indicating that pretrained transformers with NLI-based initialization are more robust to imbalanced RE data than recurrent architectures with static embeddings in these experiments.

c) Comparison with Shah et al. [37]: Using a sample of 122 hardware requirements provided by the authors, our cross-

TABLE VIII: Performance values for conflict and neutral classification task for all the requirement pair datasets. Reported values are averaged over 3 folds and results are shown as "mean ± standard deviation". Dashed (-) values indicate 0% as the performance value. The highest performance values are indicated by bold text and highlighted in blue for the Macro F1-score, and in yellow for the conflict class F1-score.

		Macro			Conflict Class		
Dataset	**Models**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
CN	TF-IDF + SVM	0.955 ± 0.001	0.931 ± 0.003	0.940 ± 0.002	0.926 ± 0.005	0.991 ± 0.003	0.957 ± 0.001
	WE(GloVe + FastText) + SVM	0.626 ± 0.223	0.522 ± 0.016	0.431 ± 0.034	0.631 ± 0.007	0.997 ± 0.001	0.773 ± 0.005
	SBERT + Linear layer	0.696 ± 0.049	0.595 ± 0.013	0.579 ± 0.012	0.672 ± 0.006	0.935 ± 0.027	0.782 ± 0.013
	deberta-base-mnli	0.994 ± 0.000	0.992 ± 0.001	0.993 ± 0.001	0.993 ± 0.000	0.999 ± 0.000	**0.996 ± 0.000**
	bert-base-uncased-MNLI	0.995 ± 0.001	0.993 ± 0.001	**0.994 ± 0.001**	0.991 ± 0.000	0.999 ± 0.000	0.995 ± 0.000
	SR-BERT	0.994 ± 0.000	0.991 ± 0.001	0.992 ± 0.001	0.989 ± 0.001	0.999 ± 0.000	0.994 ±0.000
UAV	TF-IDF + SVM	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
	WE(GloVe + FastText) + SVM	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
	SBERT + Linear layer	0.498 ± 0.000	0.499 ± 0.000	0.499 ± 0.000	-	-	-
	deberta-base-mnli	0.924 ± 0.058	0.860 ± 0.078	**0.877 ± 0.034**	0.849 ± 0.117	0.722 ± 0.157	**0.756 ± 0.068**
	bert-base-uncased-MNLI	0.876 ± 0.017	0.833 ± 0.067	0.849 ± 0.036	0.754 ± 0.035	0.666 ± 0.136	0.698 ± 0.071
	SR-BERT	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
WorldVista	TF-IDF + SVM	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
	WE(GloVe + FastText) + SVM	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
	SBERT + Linear layer	0.498 ± 0.000	0.498 ± 0.000	0.498 ± 0.000	-	-	-
	deberta-base-mnli	0.999± 0.000	0.827 ± 0.041	0.893 ± 0.031	1.000 ± 0.000	0.654 ± 0.083	0.787 ± 0.062
	bert-base-uncased-MNLI	0.978 ± 0.029	0.857 ± 0.051	**0.908 ± 0.043**	0.958 ± 0.058	0.714 ± 0.102	**0.817 ± 0.087**
	SR-BERT	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
PURE	TF-IDF + SVM	0.495 ± 0.000	0.500 ± 0.000	0.479 ± 0.000	-	-	-
	WE(GloVe + FastText) + SVM	0.495 ± 0.000	0.500 ± 0.000	0.497 ± 0.000	-	-	-
	SBERT + Linear layer	0.495 ± 0.000	0.497 ± 0.000	0.496 ± 0.000	-	-	-
	deberta-base-mnli	0.971 ± 0.039	0.928 ± 0.058	**0.948 ± 0.048**	0.944 ± 0.078	0.857 ± 0.116	**0.897 ± 0.095**
	bert-base-uncased-MNLI	0.947 ± 0.036	0.903 ± 0.030	0.919 ± 0.005	0.896 ± 0.073	0.801 ± 0.062	0.841 ± 0.011
	SR-BERT	0.495 ± 0.000	0.500 ± 0.000	0.497 ± 0.000	-	-	-
OPENCOSS	TF-IDF + SVM	0.499 ± 0.000	0.499 ± 0.000	0.499 ± 0.000	-	-	-
	WE(GloVe + FastText) + SVM	0.499 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-
	SBERT + Linear layer	0.495 ± 0.000	0.497 ± 0.000	0.496 ± 0.000	-	-	-
	deberta-base-mnli	0.721 ± 0.157	0.777 ± 0.207	0.744 ± 0.175	0.444 ± 0.314	0.555 ± 0.415	0.488 ± 0.349
	bert-base-uncased-MNLI	0.958 ± 0.058	0.791 ± 0.089	**0.841 ± 0.065**	0.916 ± 0.117	0.583 ± 0.180	**0.683 ± 0.131**
	SR-BERT	0.499 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-	-	-

TABLE IX: Evaluation of cross-domain models trained with different combinations of requirement pair datasets (Source Model). Reported values are averaged over 3 folds and presented as "mean ± standard deviations". Improved results compared to Table VIII are underlined. The highest performance in terms of Macro F1-score and Conflict-F1 is highlighted in blue. The CN-trained model utilizes the transformer checkpoint *deberta-base-mnli*, while the remaining configurations in the "Source Model" column are trained with *bert-base-uncased-MNLI*

Target Data	**Source Model**	**Precision**	**Recall**	**F1**	**Conflict-F1**
UAV	CN	0.509 ± 0.000	0.926 ± 0.002	0.478 ± 0.002	0.035 ± 0.001
	WorldVista	0.592 ± 0.002	0.966 ± 0.038	0.651 ± 0.001	0.309 ± 0.002
	PURE	0.872 ± 0.040	0.916 ±0.068	**0.892 ± 0.052**	**0.786 ± 0.105**
	OPENCOSS	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-
	WorldVista + PURE	0.876 ± 0.119	0.777 ± 0.103	0.798 ± 0.083	0.598 ± 0.166
	WorldVista + PURE + OPENCOSS	0.991 ± 0.000	0.666 ± 0.068	0.741 ± 0.078	0.484 ± 0.155
WorldVista	CN	0.527 ± 0.002	0.959 ± 0.018	0.538 ± 0.005	0.105 ± 0.009
	UAV	0.999 ± 0.000	0.702 ± 0.049	0.783 ± 0.048	0.568 ± 0.097
	PURE	0.978 ± 0.029	0.843 ± 0.036	**0.899 ± 0.035**	**0.799 ± 0.070**
	OPENCOSS	0.498 ± 0.000	0.500 ± 0.000	0.499 ±0.000	-
	UAV + PURE	0.946 ± 0.047	0.929 ± 0.017	**0.936 ± 0.031**	**0.873 ± 0.062**
	UAV + PURE + OPENCOSS	0.498 ± 0.000	0.500 ± 0.000	0.499 ± 0.000	-
PURE	CN	0.522 ± 0.002	0.904 ± 0.008	0.491 ± 0.010	0.087 ± 0.010
	UAV	0.970 ± 0.038	0.829 ± 0.084	0.876 ± 0.055	0.755 ± 0.109
	WorldVista	0.783 ± 0.048	0.996 ± 0.001	0.857 ± 0.042	0.718 ± 0.083
	OPENCOSS	0.495 ± 0.000	0.500 ± 0.000	0.497 ± 0.000	-
	UAV + WorldVista	0.998 ± 0.000	0.829 ± 0.061	0.893 ± 0.043	0.787 ± 0.085
	UAV + WorldVista + OPENCOSS	0.965 ± 0.047	0.848 ± 0.011	0.898 ± 0.025	0.797 ± 0.049
OPENCOSS	CN	0.501 ± 0.000	0.642 ± 0.001	0.223 ± 0.002	0.004 ± 0.000
	UAV	0.647 ± 0.026	0.749 ± 0.067	0.677 ± 0.015	0.355 ± 0.031
	WorldVista	0.508 ± 0.000	0.955 ± 0.002	0.492 ± 0.000	0.031 ± 0.002
	PURE	0.522 ± 0.002	0.984 ±0.001	0.535 ± 0.004	0.087 ± 0.008
	WorldVista + PURE	0.565 ± 0.006	0.940 ± 0.077	0.611 ± 0.011	0.226 ± 0.023
	UAV + WorldVista + PURE	0.515 ± 0.001	0.976 ± 0.002	0.517 ± 0.002	0.059 ± 0.005

TABLE X: Performance comparison with Malik et al. [4] (supervised semantic similarity). Reported values are averaged over 3 folds as mean ± std.

Dataset	deberta-base-mnli	bert-base-uncased-MNLI	Malik et al. [4]
WorldVista	**0.93 ± 0.01**	0.90 ± 0.03	0.86 ± 0.05
UAV	0.84 ± 0.12	**0.94 ± 0.01**	0.90 ± 0.04
PURE	**0.90 ± 0.05**	0.89 ± 0.05	0.88 ± 0.11
OPENCOSS	0.69 ± 0.14	**0.74 ± 0.17**	0.43 ± 0.04

TABLE XI: Performance comparison (Macro F1) with Abeba and Alemneh [27] (BiLSTM + Word2vec).

Approach	WorldVista	PURE	UAV
Abeba and Alemneh [27] – BiLSTM + Word2vec	0.47	0.48	0.48
bert-base-uncased-MNLI (Sequential)	**0.86**	**0.82**	**0.79**

TABLE XII: Performance comparison with Shah et al. [37] (clustering-based) on 122 hardware requirements.

Approach	Precision	Recall	F1-score
Shah et al. [37]	0.77	0.37	0.48
bert-base-uncased-MNLI (Cross-domain)	**0.83**	**0.56**	**0.60**

domain model (trained on WorldVista/UAV/PURE) achieves 0.60 F1 versus 0.48 for their clustering-based approach (Table XII). The 19-point recall gap (0.56 vs. 0.37) indicates that the cross-domain transformer recovers more conflict pairs in this sample, despite the domain difference between the training and test data.

Taken together, these comparisons show that transfer learning with NLI-pretrained transformers performs better than the evaluated rule-based feature extraction, static-embedding recurrent, and clustering-based approaches on the tested datasets. The gains are most pronounced on small or imbalanced datasets (OPENCOSS and Shah et al.'s sample) and on the dataset with more general-domain language (WorldVista), supporting the use of NLI-to-requirement transfer for this task.

E. Discussion

Figure 6 compares top-performing techniques across datasets. SR-BERT and sequential methods perform best on CDN, where the larger training set supports learning conflict and duplicate patterns. CN's binary conflict-neutral setup also supports strong in-domain performance. In contrast, SR-BERT performs poorly on WorldVista, UAV, PURE, and OPENCOSS, where conflict pairs are sparse. Sequential transfer is strongest on PURE and OPENCOSS, whereas cross-domain transfer is strongest on UAV and WorldVista. This variation suggests that transfer performance depends not only on dataset size but also on requirement style, label balance, and source-target similarity.

Overall, the results suggest a dataset-dependent modeling choice: sequential transfer is preferable for small or imbalanced datasets, SR-BERT is effective when larger labeled requirement-pair data are available, and cross-domain transfer is useful when the source and target requirements share structural similarity. Generalization therefore depends on requirement documentation style, conflict prevalence, and domain proximity between datasets.

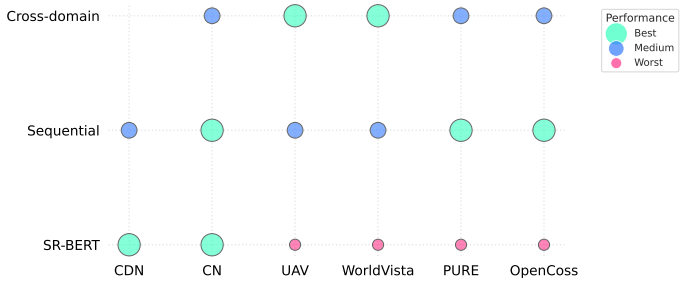


Fig. 6: Summary comparison for requirement pair datasets. 'Sequential' represents multi-stage FT. Bubble color indicates performance level, while size represents performance metric.

V. Conclusion

This study investigated transformer-based methods for conflict and duplicate detection in software requirements. We formulated the problem as requirement-pair classification, evaluated sequential and cross-domain transfer learning with pretrained transformers, and proposed SR-BERT as a domain-adapted Sentence-BERT framework for this task. The results show that SR-BERT achieves the strongest performance on the larger three-class CDN dataset, whereas sequential transfer learning with MNLI-initialized checkpoints is more effective for the smaller and more imbalanced binary conflict-neutral datasets. Cross-domain transfer is beneficial in selected source-target combinations, especially when requirement structure appears similar across datasets. Overall, the study supports sentence-pair transfer learning as a useful approach for automating requirements conflict and duplicate detection, while showing that performance remains sensitive to class balance and domain similarity. The analysis is limited to six datasets, and evaluation on additional requirement corpora would be needed to strengthen generalizability claims. The scarcity of public conflict and duplicate datasets also required modifications to existing corpora, which should be considered a threat to validity. Future work includes data augmentation for rare conflict classes, evaluation of newer transformer checkpoints, stronger entity and role extraction for requirement semantics, and broader testing in domains with similarly high-stakes requirement documentation.

References

[1] K. E. Wiegers and J. Beatty, *Software requirements*. Pearson Education, 2013.

[2] A. v. Lamsweerde, *Requirements engineering: from system goals to UML models to software specifications*. John Wiley & Sons, Ltd, 2009.

[3] W. Guo, L. Zhang, and X. Lian, "Automatically detecting the conflicts between software requirements based on finer semantic analysis," *arXiv preprint arXiv:2103.02255*, 2021.

[4] G. Malik, M. Cevik, A. Başar, and D. Parikh, "Supervised semantic similarity-based conflict detection algorithm: S3cda," in *2025 IEEE International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*. IEEE, 2025, pp. 1–10.

[5] S. Robertson and J. Robertson, *Mastering the requirements process: Getting requirements right*. Addison-wesley, 2012.

[6] K. Pohl, *Requirements engineering fundamentals: a study guide for the certified professional for requirements engineering exam-foundation level-IREB compliant*. Rocky Nook, Inc., 2016.

[7] M. Urbieta, M. J. Escalona, E. Robles Luna, and G. Rossi, "Detecting conflicts and inconsistencies in web application requirements," in *Current Trends in Web Engineering: Workshops, Doctoral Symposium, and Tutorials, Held at ICWE 2011, Paphos, Cyprus, June 20-21, 2011. Revised Selected Papers 11*. Springer, 2012, pp. 278–288.

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

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[10] F. Nazir, W. H. Butt, M. W. Anwar, and M. A. Khan Khattak, "The applications of natural language processing (nlp) for software requirement engineering-a systematic literature review," *Information Science and Applications 2017: ICISA 2017 8*, pp. 485–493, 2017.

[11] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing for requirements engineering: A systematic mapping study," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–41, 2021.

[12] R. Sonbol, G. Rebdawi, and N. Ghneim, "The use of nlp-based text representation techniques to support requirement engineering tasks: A systematic mapping review," *IEEE Access*, vol. 10, pp. 62 811–62 830, 2022.

[13] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=XPZIaotutsD

[14] S. Ruder, M. E. Peters, S. Swayamdipta, and T. Wolf, "Transfer learning in natural language processing," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 15–18. [Online]. Available: https://aclanthology.org/N19-5004

[15] T. Hey, J. Keim, A. Koziolek, and W. F. Tichy, "Norbert: Transfer learning for requirements classification," in *2020 IEEE 28th International Requirements Engineering Conference (RE)*. IEEE, 2020, pp. 169–179.

[16] D. Kici, G. Malik, M. Cevik, D. Parikh, and A. Basar, "A bert-based transfer learning approach to text classification on software requirements specifications." in *Canadian Conference on AI*, 2021.

[17] S. Ezzini, S. Abualhaija, C. Arora, and M. Sabetzadeh, "Taphsir: towards anaphoric ambiguity detection and resolution in requirements," in *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2022, pp. 1677–1681.

[18] M. Ajagbe and L. Zhao, "Retraining a bert model for transfer learning in requirements engineering: A preliminary study," in *2022 IEEE 30th International Requirements Engineering Conference (RE)*. IEEE, 2022, pp. 309–315.

[19] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *arXiv preprint arXiv:1908.10084*, 2019.

[20] W. Xu, C. Callison-Burch, and W. B. Dolan, "Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit)," in *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, 2015, pp. 1–11.

[21] B. MacCartney and C. D. Manning, "Modeling semantic containment and exclusion in natural language inference," in *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008)*, 2008, pp. 521–528.

[22] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *arXiv preprint arXiv:1704.05426*, 2017.

[23] S. Ghosh, D. Elenius, W. Li, P. Lincoln, N. Shankar, and W. Steiner, "Automatically extracting requirements specifications from natural language," *arXiv preprint arXiv:1403.3142*, 2014.

[24] M. Aldekhail and D. Ziani, "Intelligent method for software requirement conflicts identification and removal: proposed framework and analysis," *International Journal of Computer Science and Network Security*, vol. 17, no. 12, pp. 91–95, 2017.

[25] J. Polpinij and A. Ghose, "An automatic elaborate requirement specification by using hierarchical text classification," in *2008 International conference on computer science and software engineering*, vol. 1. IEEE, 2008, pp. 706–709.

[26] T. Viana, A. Zisman, and A. K. Bandara, "Identifying conflicting requirements in systems of systems," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*. IEEE, 2017, pp. 436–441.

[27] G. Abeba and E. Alemneh, "Identification of nonfunctional requirement conflicts: Machine learning approach," in *International Conference on Advances of Science and Technology*. Springer, 2021, pp. 435–445.

[28] J. He, L. Xu, M. Yan, X. Xia, and Y. Lei, "Duplicate bug report detection using dual-channel convolutional neural networks," in *Proceedings of the 28th International Conference on Program Comprehension*, 2020, pp. 117–127.

[29] L. Wang, L. Zhang, and J. Jiang, "Duplicate question detection with deep learning in stack overflow," *IEEE Access*, vol. 8, pp. 25 964–25 975, 2020.

[30] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," *IEEE Transactions on Software Engineering*, vol. 39, no. 1, pp. 18–44, 2011.

[31] D. Rao, L. Bian, and H. Zhao, "Research of duplicate requirement detection method," in *Smart Computing and Communication: 7th International Conference, SmartCom 2022, New York City, NY, USA, November 18–20, 2022, Proceedings*. Springer, 2023, pp. 213–225.

[32] J. Dick, E. Hull, and K. Jackson, "Doors: A tool to manage requirements," in *Requirements Engineering*. Springer, 2017, pp. 187–206.

[33] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (ears)," in *2009 17th IEEE International Requirements Engineering Conference*. IEEE, 2009, pp. 317–322.

[34] A. Ferrari, G. O. Spagnolo, and S. Gnesi, "Pure: A dataset of public requirements documents," in *2017 IEEE 25th International Requirements Engineering Conference (RE)*, 2017, pp. 502–505.

[35] Z. Cao, Y. Zhou, A. Yang, and S. Peng, "Deep transfer learning mechanism for fine-grained cross-domain sentiment classification," *Connection Science*, vol. 33, no. 4, pp. 911–928, 2021.

[36] K. Aderghal, A. Khvostikov, A. Krylov, J. Benois-Pineau, K. Afdel, and G. Catheline, "Classification of alzheimer disease on imaging modalities with deep cnns using cross-modal transfer learning," in *2018 IEEE 31st international symposium on computer-based medical systems (CBMS)*. IEEE, 2018, pp. 345–350.

[37] U. Shah, S. Patel, and D. C. Jinwala, "Detecting intra-conflicts in non-functional requirements," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 29,

no. 03, pp. 435–461, 2021.