Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

Data-efficient classification of road inspection texts with a semantic similarity criterion

Ching Yau Fergus Mok *, Lavindra de Silva, Varun Kumar Reja, Stephen Green, Ioannis Brilakis

Department of Engineering, University of Cambridge, Cambridge, United Kingdom

ARTICLE INFO

Keywords:
Natural language processing
Large language model
Text classification
Few-shot learning
Semantic similarity

ABSTRACT

Road maintenance involves manually classifying a large volume of textual data necessary for downstream applications such as raising a maintenance job order. Automation can not only bring significant time and cost savings, but it can also facilitate digitalization efforts like the Road Digital Twin (DT). However, as is the case with many Architecture, Engineering and Construction (AEC) applications, annotated data availability is low, which demands exploration of specialized techniques for resource-constrained settings that have not been focused on in engineering. This work bridges this gap by proposing a data-efficient similarity-based text classifier that aims at effectively utilizing existing domain knowledge and pre-training knowledge of Large Language Models (LLMs) to enable rapid domain adaptation. It reformulates text classification as a similarity comparison task, using semantics directly as a classification criterion. Through a case study on classifying road inspection comments, the proposed classifier outperformed both traditionally fine-tuned and few-shot learning approaches. It attained an $f1$ score of 0.46 with just one example per class, equivalent to the value for Sentence Transformer Fine-Tuning (SetFit) with 4 examples and *Llama3* with 10. Additionally, it is able to keep up with traditional fine-tuning methods when trained with more than 300,000 total examples, achieving an $accuracy$ of more than 95% and $f1$ of around 0.9. These results indicate that the proposal is competitive against traditionally fine-tuned and few-shot models across all levels of data availability. This versatility significantly elevates the feasibility of deploying an automated text classification pipeline in a complex engineering field like road maintenance.

1. Introduction

Road maintenance generates a large amount of textual data that needs to be manually analyzed. One primary source is from manual visual inspections of road assets where inspectors record defects and other anomalies in unstructured textual fields. These fields are crucial in that they capture the specifics of an entry that are required for further downstream processes like raising a maintenance job order. Texts are classified according to inspection manuals and other documentation which define key asset data requirements. In the UK, documents such as the Asset Data Management Manual (ADMM) [1] outline these standards. Another emerging downstream process is data integration with digital platforms. National Highways in the UK is proposing the creation of a Road Digital Twin (DT) [2] which encapsulates key road asset data in a digital replica of the road network. As a part of the Road DT data requirements, detailed information about road assets are required [3]. To fulfill the information needs of these processes, structured information extraction from these comments is necessary. However, manual assessment is time consuming and unscalable, especially considering the increasing need for systematic information extraction due to digitalization efforts. This forms the motivation for this work, which explores techniques that can be used to automate the analysis of textual road inspection comments.

Automated text classification is a means of satisfying downstream data requirements in a more scalable manner. It is an application of natural language processing (NLP) which involves using a computational model to assign a label from a pre-defined list of classes to a sequence of words. It has advanced significantly in recent years as a result of the development of powerful Large Language Models (LLMs) like Generative Pre-Trained Transformers (GPT) [4] and Bidirectional Encoder Representations from Transformers (BERT) [5]. These models possess task-agnostic knowledge through large-scale pre-training on general context texts, and task adaptation is commonly performed through fine-tuning with a comparatively smaller set of in-domain training data.

* Corresponding author.
E-mail addresses: cym23@cam.ac.uk (C.Y.F. Mok), lpd25@cam.ac.uk (L. de Silva), varunreja7@gmail.com (V.K. Reja), slg79@cam.ac.uk (S. Green), ib340@cam.ac.uk (I. Brilakis).

https://doi.org/10.1016/j.aei.2025.103509
Received 3 January 2025; Received in revised form 6 May 2025; Accepted 26 May 2025
Available online 14 June 2025

Adapting general models for domain-specific applications can be challenging if there is insufficient annotated training data. This is the case in the Architecture, Engineering and Construction (AEC) domain, including road maintenance, where domain-specific training data are often unavailable, private, or lack in quality [6]. Improving the resource-efficiency of adaptation techniques can help tackle this deficiency. Though there has been research on this in more general settings, such as with few-shot learning techniques or in-context learning, it has not been an area of focus in NLP research in the AEC domain. Particularly, resource-efficient fine-tuning and more effective existing domain knowledge utilization have not been explored in the AEC domain. This is especially critical given the lack of data that is so prevalent in the field.

The authors propose to bridge this gap by introducing a resource-efficient text classification pipeline for road inspection texts which utilize existing domain knowledge to effectively guide models to quickly adapt to new tasks. The classifier works by first embedding both a test input and descriptive textual class definitions (taken from existing sources of domain knowledge like inspection manuals) in a common embedding space using Sentence BERT (SBERT), followed by cosine similarity comparisons between pairs of input and class descriptions, selecting the pair with the highest score as the prediction. It leverages *SBERT*'s ability to output semantically representative vectors, allowing for simple similarity calculations between sentences. Resource-efficiency in fine-tuning is achieved in two ways: Firstly, the incorporation of existing domain knowledge through the use of class descriptions can effectively infuse task information to the classifier in a complementary way to training examples. Secondly, by aligning the classification objectives with the contrastive similarity-based pre-training objectives of *SBERT*, pre-training knowledge can be fully utilized, allowing for in theory zero-shot (no fine-tuning) applications. In this work, text classification is remodeled as a similarity comparison task between an input and class definitions.

This classifier differs from related works in the following ways: Firstly, descriptive textual class definitions are used to represent classes as opposed to uninformative indices. These descriptions act to explicitly infuse existing domain knowledge from sources such as inspection manuals or standards documentation to allow the model to infer class contents, as opposed to solely relying on training examples. Secondly, semantic similarity between an input and class definitions is directly used as the classification criterion as opposed to softmax outputs for neurons in an uninitialized classification head, removing the need for task-dependent changes to the classifier architecture that does not leverage pre-training knowledge. Together, these two distinctions lessen the burden of supplying task information through training examples by introducing an additional layer of knowledge in the form of descriptive class definitions, enabling resource-efficient fine-tuning of models.

To the best of the authors' knowledge, this work is the first to contribute in the following areas:

1. Classifying road maintenance (and wider AEC) texts with a focus on resource-efficiency in model adaptation through maximal domain knowledge and pre-training knowledge utilization.
2. Fine-tuning a similarity-based model, specifically SBERT, for a single-text classification task with direct use of classification training objectives.

This proposal is validated through a case study on classifying road-work inspection comments according to road code violations in New York City (NYC) using a public data from the NYC Open Data. The cleaned dataset contains more than 400,000 entries, and 83 classes. Results from various resource-constrained settings indicate that the proposed classifier outperformed competing techniques when data availability is low, but still exhibits comparable performance when no constraint is present. This indicates that the model is versatile and resilient against the level of data availability, which is beneficial for many engineering applications where data abundance across classes vary significantly.

The paper is structured as follows: Section 2 provides an overview of the state of research in the area in addition to background knowledge for techniques used in the experiments. Section 3 outlines architectures and fine-tuning regimes for the propose classifiers and elaborates on how they differ from existing approaches. Section 4 details the case study used to answer the research questions set out which simulates the task of automatically classifying road maintenance comments. Section 5 highlights the results of the case study, including validating the performances of the proposed classifiers and comparing them with alternative baseline approaches. Section 6 further consolidates the observations identified by performing additional experiments. Section 7 discusses the limitations to this work along with extensions proposed to overcome them. Finally, Section 8 summarizes the paper and discusses its implications for road maintenance and the wider AEC context.

2. Background and related work

This section explores the motivation for this work along with other research relevant to this paper. Section 2.1 explores the state of practice in road inspection and maintenance, which forms the motivation for this work. Section 2.2 explores existing applications of NLP in the AEC domain and the common difficulties faced during their implementation. Section 2.3 provides a background for LLMs which caused a paradigm shift in NLP, along with a detailed explanation of a relevant model, SBERT. Section 2.4 discusses data-efficient techniques that are used mostly in other fields that can serve as inspirations for a proposed pipeline. Finally, 2.5 summarizes the research gaps identified along with the research questions set out.

2.1. Road inspection and maintenance

Road maintenance is a major responsibility for road authorities. In the UK, National Highways manages the Strategic Road Network, which carries a third of all traffic and two-thirds of all freight [7]. Though limited automated inspection techniques exist [8], most roads are still inspected manually through coarse and detailed visual inspections [9]. For National Highways, inspectors record irregularities through a road asset management database called CONFIRM, where each entry consist of high level structured fields such as time and defect type, and free text fields which denote more specific information not captured by predefined categories in the system. These free text fields contain crucial data requirements for downstream processes written in unstructured natural language and often with heavy use of technical jargons.

An example of a downstream application is raising a maintenance job order, where maintenance engineers have to analyze each entry to determine the optimal repair regime, akin to other domains such as building maintenance [10]. Documents such as the ADMM or asset-specific manuals such as the Traffic Sign Manual [11] and Maintenance of Traffic Signs [12] are used as reference for what structured information need to be extracted. They contain descriptive annotations of possible defect categories and properties. For example, the Traffic Sign Manual specifies that “Excessively discolored or faded signs (e.g. white backgrounds that have become grey or brown, or red borders faded to pink) and signs where the legend or graphic is peeling are not effective and need to be replaced”.

Another emerging downstream process is the fulfillment of information requirements of the proposed Road DT [2]. As with digital twins in other smart infrastructure, an Information Delivery Manual (IDM) specifies the information required at various stages of an operational process [13,14]. For example, [3] outlined the information required for instances of roadside vegetation overgrowth, and it includes defect details such as qualitative descriptions about its appearance and specific dimensional measurements.

Due to the costly nature of manual analysis of these unstructured comments, there are considerable benefits in adopting automated techniques. A systematic and reproducible analysis of these textual data can unlock knowledge that can be beneficial to the industry, in terms of more efficient data retrieval and more informed decision making and planning. This is particularly pressing due to the push for digitization of road asset management with the Road DT. There is a lot of potential in how this can be achieved in a resource-efficient and scalable manner given that there are already numerous sources of domain knowledge in manuals, guidance, and the IDM. Leveraging these sources can significantly ease the automation process of structured information extraction to fulfill downstream data requirements in road maintenance.

2.2. NLP in the AEC domain

Road maintenance is an example of AEC operations, where textual data are often more complex than those from a general context, complicating the implementation of NLP pipelines. For instance, the use of domain-specific vocabularies and the diversity of possible categorizations can limit the capabilities of general techniques. Due to the task and domain-specific nature of many AEC texts, there is a large body of research that explores the use of tailored NLP techniques that can leverage them in downstream use cases. For example, [15] developed a multi-label classification model for texts covering flash flood events using a fine-tuned BERT to improve the organization of records in natural disaster databases. [16] designed an automated system for construction specification review using NLP techniques such as named entity recognition (NER) with Bi-LSTM and document embedding through Doc2Vec to streamline processes and improve construction risk management. [17] built classifiers for maintenance logs at a manufacturing facility and demonstrated that fine-tuned LLMs (CamemBERT) can outperform traditional techniques such as Term Frequency - Inverse Document Frequency (TF-IDF) and random forests even in domain-specific setting.

Though these work demonstrated the potential of adopting modern NLP techniques in the AEC domain, they have yet to tackle the following issues that are unique to this field:

2.2.1. Capturing the complexity of AEC applications

One aspect of AEC applications that has not been fully captured by prior research is the complexity in the NLP tasks. For instance, many text classification tasks in related works only involve a small number of relatively general classes: [15] classified texts into one of seven categories associated with different flood-related information such as “Damage and Economic Impact”. [17] classified maintenance texts into one of two disturbance categories and one of five workload categories.

Real world AEC applications are often stringent and tightly regulated. Along with the complexity and precision of the task, this can lead to a large number of technical classes defined by strict criteria [3]. Instead of classifying whether an inspection comment indicates a defective asset, the task would be more akin to classifying the specific defect type. For instance, road defects alone can be subdivided into classes such as longitudinal cracks, transverse cracks, alligator cracks, potholes, rutting, and delamination. Similarly, drainage-related issues span categories like blocked culverts, sedimented catch basins, and channel erosion, while vegetation management addresses overgrown foliage, obstructive branches, and invasive species. These categories are often more fine-grained than those found in other works, which might pose a challenge to unadapted classifiers.

2.2.2. Tackling data scarcity

Another aspect that has not been focused on is tackling the fundamental issue regarding data scarcity that is widespread in the field [6]. [15] used a dataset containing 21,180 paragraphs, [16] trained the NER model with 4659 sentences, and [17] used a dataset with 7000 training examples.

The lack of available data combined with the intensive data needs to train models motivated the development of Technical Language Processing (TLP) [6], an NLP framework which focuses on improving data access and quality in the AEC domain. To evaluate how TLP can be utilized in a real world setting, [18] undertook an experiment which investigated the level of agreement by experts on classifying failure events of equipment related to oil and gas. Results indicate that it is difficult to classify and tag entries consistently due to category overlap or lack of sufficient context. Noting the significant amount of human effort required to fully label datasets, [19] proposed a hybrid manual-automated approach to classify maintenance requests in buildings which prioritized words to be annotated according to frequency and statistical importance. These studies demonstrate the continuing difficulties in curating high quality, high volume datasets for AEC tasks, despite efforts such as TLP.

The inadequacies in data resources make it difficult for AEC practitioners to scalably build NLP pipelines to automate technical textual analysis processes. This highlights the need to consider alternative, resource-efficient approaches. One recent work proposed a few-shot named entity recognition model for information extraction from bridge inspection texts [20]. However, data-efficient methods have yet been fully explored in the AEC domain.

2.3. LLMs

LLMs have large potentials in contributing to improving resource-efficiency in automating the analysis of AEC texts. They are built on transformer models [21] which utilizes the attention mechanism [22] to model dependencies as opposed to neural connections in the case of RNN and LSTM, allowing them to effectively process long sequences. LLMs are transformer models that were pre-trained over large corpora of texts, enabling them to acquire significant task-agnostic knowledge. For example, *BERT* was pre-trained over 800 million words from BooksCorpus and 2500 million words from the English Wikipedia [5]. As a result, they can be easily adopted in many use cases with relatively minor adjustments. Adaptation of LLMs to specific domains and tasks is typically done through fine-tuning, which is a transfer learning technique where a comparatively smaller, in-domain dataset is used to adapt model parameters.

Many models have been developed that differ in size, pre-training objectives, and performance. Popular open-source LLMs include *Llama* and *Mistral* models, and smaller models such as *BERT* are also often termed Pre-trained Language Models. In the context of this work, all pre-trained transformer-based models are considered as LLMs. It is worth noting that a common issue that faces LLM applications is hallucination, defined as “generated content that is either nonsensical or unfaithful to the provided source content”. [23]. However, as LLM-based classifiers do not generate texts, this source of error does not apply.

2.3.1. SBERT

A particular model of interest for this work is *SBERT* [24] as it excels in semantic textual similarity tasks. It was built on *BERT* and fine-tuned using a contrastive learning training objective, where pairs of texts were input to produce an output that is indicative of their relationship. For example, the output can indicate whether one sentence entails the another. It is a popular model for tasks where similarity between two inputs is required, like an information retrieval task where it is used to select candidate passages according to a query using similarity comparisons.

In the context of this work, the authors hypothesize that it can be used to effectively compute similarities between an inspection text and its classification criterion formulated as descriptive class definitions taken from sources of domain knowledge, such as inspection manuals and standards documentation. This was built on the assumption that inspection texts are semantically similar to the descriptive definition

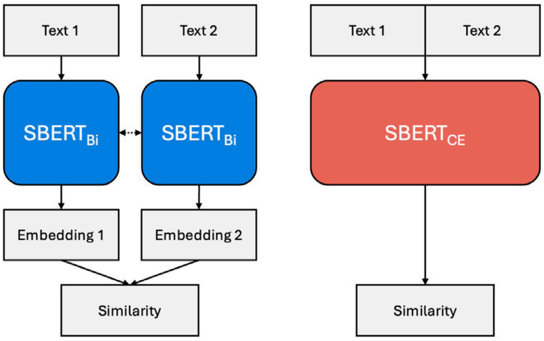


Fig. 1. Comparison between Bi-Encoders and cross Encoders [25].

of classes that they belong to in a classification task. As similarity calculation does not require the use of an uninitialized classification head, this enables in theory zero-shot applications and rapid domain adaptation in few-shot settings.

To fully leverage *SBERT* for this purpose, the two main configurations of *SBERT* are explored: Bi-Encoder and Cross Encoder. An overview is illustrated in Fig. 1. The aim of both types of *SBERT* is to output similarity scores of sentence pairs. The difference between them lies in the intermediary steps in computing this similarity which results in differing computational requirements at inference, being different functions of S and C, where S is the number of testing sentences and C is the number of classes.

Bi-Encoder Bi-Encoders compute a static embedding for each sentence independently using the same encoder, meaning that each sentence is only required to be encoded once. The embedding correlates to the semantic content of the input, allowing distances between embeddings to indicate semantic similarity. Distances can be calculated in a cost-effective way through simple vector metrics, such as cosine similarity.

This approach is efficient as the number of encodings at inference is simply $S + C$. However, as the encoding of an input is static (independent of the class it is compared against), flexibility and accuracy are sacrificed.

Cross encoder Cross Encoders directly output similarities without intermediary embeddings. Every input-class permutation must be encoded separately. They are computationally more expensive, as the number of encodings is $S \times C$. This is especially problematic when the number of classes is large. They tend to exhibit better performances [24] due to the transformer's ability to pay cross attention between the input and the class, offering more flexible encodings at the expense of computational efficiency.

2.4. Data-efficient model adaptation

As mentioned in Section 2.3, a common way of adapting LLMs for domain-specific applications is through fine-tuning, an example of transfer learning where knowledge from related training data is transferred to a downstream application. However, fine-tuning alone does not achieve a high degree of data efficiency as evident by the large volume of training data used in [15,17]. Therefore, other techniques that can be used in place of or alongside fine-tuning are explored. It is worth noting that these techniques have not been focused on in NLP research in AEC.

2.4.1. Traditional few-shot classification

Most traditional few-shot learning (FSL) techniques utilize a "support set" to supply class examples to a "query set" corresponding to the inputs. They are commonly trained episodically where the model learns to relate the support and query sets of each training class in turn. At inference, only the support sets of the target classes which were not seen during training are given, with which the query set is compared against.

There are approaches that focus on embedding inputs effectively to enable the use of simple classifiers, termed metric-based meta-learning. The Matching Network [26], the Prototypical Network [27], and the Relation Network [28] are examples of this, with the differences between them lying in how classes are embedded and hoe distances are calculated. Dopierre et al. [29] adapted these approaches to the transformer architecture and compared their results to a baseline, traditionally fine-tuned BERT model. Results indicated that Prototypical Network outperformed other models, though the improvement was not significant over the baseline.

There are also model-based meta-learning approaches where models adapt to new tasks by modifying the neural networks themselves. Conditional Neural Processes [30] and Conditional Neural Adaptive Processes [31] are examples where parameters of the neural network are tweaked at inference. [32] adapted the Conditional Neural Adaptive Processes framework for a text classification task with transformers, using gradient information to represent tasks.

Despite being able to quickly adapt to new, unseen classes when only given a small number of support examples, episodic training still requires a large amount of related training data. The idea is that although task-specific data is not widely available, data that is distantly relevant are. This is not always valid in the AEC domain, as tasks are often unique and there is limited transferable knowledge between applications. Sentence Transformer Fine-Tuning (SetFit) [33] has been proposed to train classifiers in a few-shot setting without the use of close-domain data and prompts, solely relying on few-shot training examples. It works by contrastively fine-tuning SBERT to distinguish between examples belonging to the same class and those belonging to different classes, allowing it to efficiently embed inputs. Classification is then carried out with a simple neural network as with other metric-learning approach. [34] implemented SetFit to automatically score essays, while [35] compared SetFit's performance with large-scale and proprietary LLM models such as GPT 3.5 in a financial text classification task.

Although traditional FSL techniques has been widely researched, they have not been focused on in the AEC domain. Moreover, they solely rely on training examples to adapt models, with no clear means of integrating existing domain knowledge from sources such as classification guidance and standards into the classification process.

2.4.2. In-context learning

Another technique for model adaptation is through prompting. In-context learning (ICL) involves conditioning the model's output through the provision of a prompt containing the context of a classification task [36]. The context can involve a list of possible classes and their associated few-shot examples. References to classes can include descriptive sentences about class contents, allowing the infusion of domain knowledge into the adaptation process. As the conditioning occurs at the inference stage, no fine-tuning is required.

One problem with ICL is that most applications use larger LLMs (>7 billion parameters [37]), as they require both a comprehension of the few-shot examples and of the task formulation. The requirement of large models can be problematic when there is computational power constraints, such as the availability of graphics card memory. [38] examined the performance of ICL across various domains such as news and medical texts. It was shown that in few-shot scenarios, ICL had marginal improvements over traditional fine-tuning techniques, despite using much larger models.

Another challenge of ICL is the limited context window available [37]. The context window is effectively the maximum length of text that a model can process at once, which is fixed for each pre-trained model. This static size can become an issue when there is a large number of classes for which examples need to be supplied in one prompt. It has been shown that this issue is amplified when using smaller models [37]

2.4.3. Similarity-based classification

Similarity-based approaches to classification have also been proposed for low-resource use cases. They work by explicitly computing similarity scores between an input and a set of textual class representations, and selecting the pair with the highest score as the prediction. Class representations can be any textual descriptions of a class: [39] used label names, [40] used enriched expert defined keywords, and [41,42] also using manually defined keywords. In order to calculate similarity scores, semantics-informed representations of textual inputs are required. A popular technique to perform this is to embed class descriptions in the same feature space as the textual inputs, for example through *SBERT*, where the class descriptions act as a rough location of where the class should be in the common feature space.

A key distinction between these approaches and SetFit is that they directly use similarity as a classification criterion, whereas SetFit only uses them as a training objective. Similarly to ICL, the class definitions can be used to directly infuse domain knowledge into the classifier. However, by remodeling the classification task as a similarity-comparison task, similarity-based classifiers do not suffer from the task ambiguity caused by prompting, allowing the use of much smaller models. For example, typical SBERT models have roughly 100 million parameters. In addition, as similarity comparisons happen sequentially as opposed to encapsulating all task information in a single prompt, they do not suffer from the context window constraint in the case of having many classes.

Many of these approaches target a zero-shot scenario where no training is performed. Veeranna et al. [39] proposed the use of a similarity-based classification model to supplement a traditionally trained model to perform multi-label classification (each input having multiple labels), with the similarity-based model used to classify unseen labels through semantic similarity comparisons with label names in a 0-shot fashion. Haj-Yahia et al. [40] suggested using a set of enrichment steps to enhance manually generated label descriptions, such as adding synonyms to label keywords. Schopf et al. [41] proposed Lbl2Vec, which is a framework for similarity-based document classification that makes use of candidate examples of a class to adjust the class centroid in the vector space, with an initial guess being the vector embedding of manually generated topic keywords. The same authors expanded Lbl2vec to use LLMs such as *BERT*, and compared this approach to using other baseline methods, such as an un-finetuned *SBERT* [42].

These techniques can be transferred to a few-shot case through fine-tuning the encoder, though there are no established practices in performing this for a single-text classification task (as opposed to a paired classification task which determines the relationship between a pair of texts). *SBERT* is commonly fine-tuned in a contrastive manner [33,43] where pairs of inputs and labels indicating their pairwise relationships are provided to train the model to distinguish between similar and dissimilar inputs, unlike single-text classification tasks are typically trained using cross entropy with class logits. *SBERT* is primarily fine-tuned for paired classification tasks like predicting paraphrasing, duplication, or entailment, which differs from standard, single-text classification tasks where the focus is on categorizing a single input (e.g. classifying a road inspection comment into pre-defined categories).

One indirect way of using SBERT and pairwise similarity in a single-text classification task is to use already-labeled instances as comparison targets, and select the categorization based on the labels of the most similar ones. [44] performed this for a patent classification task using a contrastively fine-tuned SBERT via cosine similarity. Another way that SBERT can be used is to forgo similarity entirely and simply use it as a text encoder just like other LLMs such as standard BERT [45]. However, to the best of the authors' knowledge, fine-tuning SBERT directly as a similarity-based classifier with single-text classification objectives as opposed to pairwise similarity objectives has not been explored.

2.5. Research gaps and questions

Through the literature covered above, the following key research gaps have been identified:

Gap 1: The analysis of road maintenance texts is predominantly manual, limiting their scalability.

Gap 2: Improving the data-efficiency of training AEC text classifiers has not been focused on, particularly for complex applications with many classes.

Gap 3: Despite the ability to handle applications with many classes (unlike ICL), fine-tuning similarity-based classifiers has not been fully explored.

The authors identified that automating the process of analyzing road maintenance texts can contribute to improving the effectiveness of road management and be beneficial to digitalization efforts such as the Road DT. Given the difficulty in acquiring sufficient training examples in many AEC applications, the data-efficiency of automation pipelines must first be improved in order to ensure a scalable solution. The authors hypothesize that this can be achieved by employing a similarity-based text classifier that does not suffer from the drawbacks of other data-efficient approaches, such as the limited context window size in ICL. As a result, the following research questions are raised:

RQ1: Can semantic similarity with descriptive class definitions be used as a classification criterion for a road maintenance (and wider AEC) text classifier in a complex application with many technical classes?

RQ2: How should the similarity-based classifiers be structured to effectively process road maintenance texts and their associated class representations?

RQ3: How can similarity-based classifiers be fine-tuned with limited availability of road maintenance texts?

RQ4: How do similarity-based techniques compare with other traditional and resource-efficient techniques in terms of training data needs and performance on classifying road maintenance texts?

This paper aims at bridging the research gaps and answering the research questions raised.

3. Proposed classifier

The authors propose a similarity-based text classifier with *SBERT* that can effectively leverage both pre-training and existing domain knowledge to enable resource-efficient adaptation to specialized domains, from road maintenance to the wider AEC field. The classifier works by aligning a textual input with descriptive class definitions to produce similarity scores, and the pair with the highest score is selected as the prediction. Class definitions can be taken from domain knowledge documentation that outlines classification criteria, allowing existing knowledge to be efficiently infused in the classification pipeline. Two classifier architectures are proposed using the two main configurations of *SBERT*. In addition, as fine-tuning a similarity-based model for classification has not been fully explored, two model adaptation techniques are proposed. Detailed information about the classification and resource-efficient fine-tuning algorithms are included in the following sections.

The proposed classifier is distinct from other text classification AEC works (Section 2.2) in that there is an emphasis of maximal knowledge utilization from both existing domain information and the pre-training stage, allowing for rapid domain adaptation and resource-efficiency in fine-tuning. Specifically:

1. Classes are explicitly represented informatively through descriptive class definitions as opposed to randomly indexed neurons in an uninitialized classification head, allowing for class content to be pre-defined through existing domain knowledge.

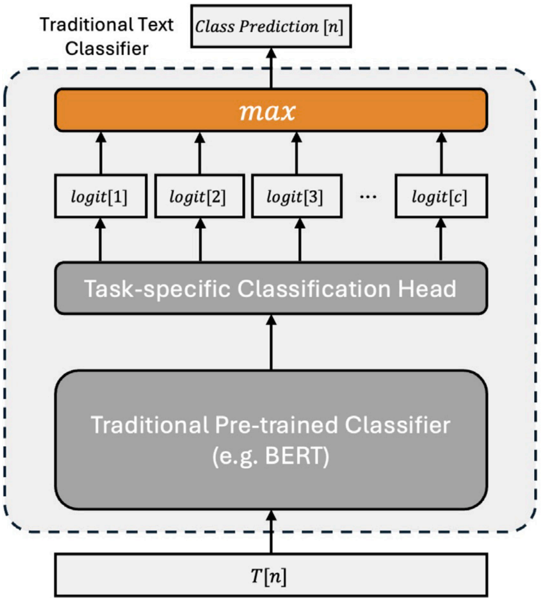

Traditional Text Classifier
Class Prediction [n]
max
logit[1]
logit[2]
logit[3]
···
logit[c]
Task-specific Classification Head
Traditional Pre-trained Classifier (e.g. BERT)
T[n]

Fig. 2. Illustration of a traditional classifier for the nth test input $T[n]$ and classes 1 to c represented as neurons in the classification head.

2. Semantic similarity is directly used as a classification criterion as opposed to softmax outputs of model logits, effectively remodeling text classification as a similarity comparison task to fully leverage the linguistic knowledge of *SBERT* gained during pre-training.

Additionally, the proposed classifier contributes to research in resource-efficient text classification (Section 2.4) in the following aspects:

1. *SBERT* is explicitly used as a fine-tuned classifier for single-text inputs, with the direct use of single-text classification training objectives as opposed to pairwise similarity objectives.
2. Different *SBERT* architectures for classification are explored to leverage both the Bi-Encoder and Cross Encoder configurations.

The following sections will outline the proposed architectures for the similarity-based classifier and the classification-focused fine-tuning regimes.

3.1. Architecture

As detailed in Section 2.3.1, the two main configurations of SBERT are the Bi-Encoder and Cross Encoder. Bi-Encoders are less accurate but the number of encoding operations scale linearly with the number of input examples and classes, whereas Cross Encoders are more accurate but scale quadratically. In order to leverage the functionalities of both configurations, two architectures are proposed. The first utilizes the Bi-Encoder as the sole classifier, and the second uses the Bi-Encoder to select a small number of candidates for a Cross Encoder reranking classifier. Both architectures do not utilize uninitialized parameters found in traditional classifiers with task-specific classification heads illustrated in Fig. 2, which are usually fully connected networks where each neuron represents each class. This means that the architecture is consistent for all tasks, allowing for in theory zero-shot applications and rapid few-shot domain adaptation by fully leveraging pre-training knowledge. The two proposed architectures are now described in detail.

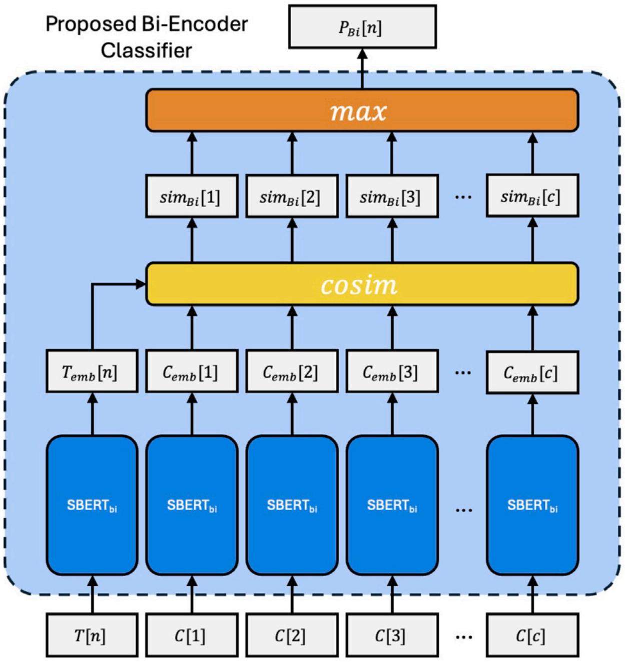


Fig. 3. Illustration of the proposed Bi-Encoder similarity-based classifier for the nth test input $T[n]$ and classes $C[1:c]$.

Algorithm 1: Bi-Encoder Only Classifier

```
Input: $C$: List of Descriptive Class Definitions,
$T$: List of Texts to Classify
Output: $P_{Bi}$: Bi-Encoder Classification Predictions
$c = |C|$;
$t = |T|$;
// Pre-embed class definitions and texts
$C_{emb} \leftarrow \mathrm{SBERT}_{Bi}(C)$;
for $j = 1$ to $t$ do
    // Embed the $j$th text
    $T_{emb}[j] \leftarrow \mathrm{SBERT}_{Bi}(T[j])$;
    // Compute similarity with all classes
    $sim_{Bi}[1:c, j] \leftarrow \mathtt{cosim}(C_{emb}[1:c], T_{emb}[j])$;
    // Find class with maximum similarity
    $P_{Bi}[j] \leftarrow \mathtt{max}_i(sim_{Bi}[i, j])$
end
return $P_{Bi}$
```

3.1.1. Bi-Encoder Only classifier

The first architecture simply uses the Bi-Encoder as the sole classifier, illustrated in Fig. 3. A notable difference between this and traditional classifiers (Fig. 2) is that there is no uninitialized classification head and class information is represented as independent inputs. Additionally, logits are replaced with similarity scores for classification. The classification process follows Algorithm 1. First, all descriptive class definitions C are encoded using a fine-tuned *SBERT* Bi-Encoder model to produce high dimension embeddings C_{emb}. Then, for each testing input $T[i]$, the corresponding embedding $T_{emb}[i]$ would be compared with every pre-computed class embedding C_{emb} through cosine similarity, defined by Eq. (1) where D represents the embedding dimension. Finally, the pair with the highest value for each input i is selected as the prediction $P_{Bi}[i]$.

$$\mathrm{cosim}(\mathbf{C_{emb}[i]}, \mathbf{T_{emb}[j]}) = \frac{\sum_{d=1}^{D} C_{emb}[i]_d T_{emb}[j]_d}{\sum_{d=1}^{D} \sqrt{C_{emb}[i]_d^2} \sum_{d=1}^{D} \sqrt{T_{emb}[j]_d^2}} \tag{1}$$

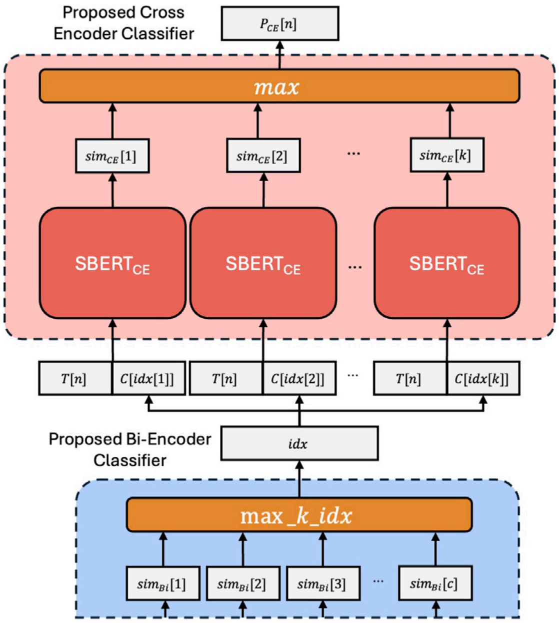


Fig. 4. Illustration of the proposed Cross Encoder similarity-based classifier for the nth test input $T[n]$ and classes $C[1 : c]$.

Cosine similarity was chosen as it is commonly used in other related works such as [44]. However, preliminary investigations using different distance metrics showed similar performances, indicating that the choice of the specific metric likely will not significantly impact performances.

3.1.2. Reranked Cross Encoder Classifier

```
Algorithm 2: Reranked Cross Encoder Classifier
  Input: $C$: List of Descriptive Class Definitions,
  $T$: List of Texts to Classify,
  $k$: Number of Bi-Encoder Candidates
  Output: $P_{CE}$: Cross Encoder Classification Predictions
  $c = |C|$;
  $t = |T|$;
  // Pre-embed class definitions
  $C_{emb} \leftarrow \mathrm{SBERT}_{Bi}(C)$;
  for $j = 1$ to $t$ do
      // Embed the jth text
      $T_{emb}[j] \leftarrow \mathrm{SBERT}_{Bi}(T[j])$;
      // Compute similarity with all classes
      $sim_{Bi}[1:c, j] \leftarrow \mathtt{cosim}(C_{emb}[1:c], T_{emb}[j])$;
      // Get indices of top k most similar classes
      $idx[j, 1:k] \leftarrow \mathtt{max\_k\_idx}_i(sim_{Bi}[i, j])$;
      // Compute CE similarities for candidates
      $sim_{CE}[1:k, j] \leftarrow \mathrm{SBERT}_{CE}(C[idx[j, 1:k]], T[j])$;
      // Find class with maximum similarity
      $P_{CE}[j] \leftarrow \mathtt{max}_n(sim_{CE}[n, j])$;
  end
  return $P_{CE}$
```

The second architecture proposed involves reranking the Bi-Encoder retrieval outputs with a Cross Encoder. Reranking is a popular concept in NLP Information Retrieval tasks [46], where a faster but less accurate model is used to "retrieve" a small number of candidates to be "reranked" by a slower but more powerful model, finally producing a single output leveraging the capabilities of both models in terms of speed and accuracy.

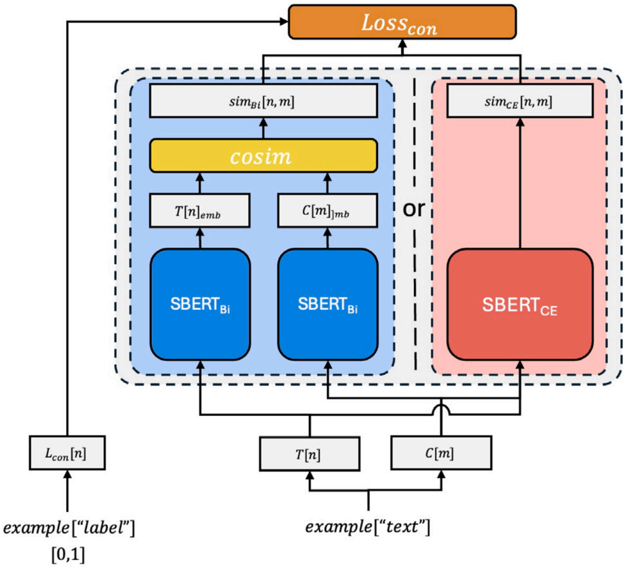


Fig. 5. Illustration of the proposed contrastive fine-tuning process for both the Bi-Encoder (left) and Cross Encoder (right) configurations.

Fig. 4 illustrates the Cross Encoder classifier that leverages the similarity outputs of the Bi-Encoder model. Algorithm 2 describes the classification process for the proposed reranking classifier. The start of the process is similar to the Bi-Encoder only approach, but instead of selecting the pair with the highest cosine similarity for each input j, the indices of the top k classes $idx[j, 1 : k]$ are stored, where k is a pre-defined number of candidates. The corresponding candidate classes $C[idx[j, n]]$ are then reranked by the Cross Encoder and the pair with the highest similarity is chosen as the prediction $P_C E[j]$.

3.2. Fine-tuning

As described in Section 2.4.3, SBERT is traditionally fine-tuned contrastively for pairwise similarity-related tasks such as paraphrasing prediction between paired inputs. Fine-tuning SBERT for tasks that do not require pairwise classification or similarity, such as categorizing road inspection comments, has not been focused on to the best of our knowledge. As a result, two fine-tuning strategies are proposed for this purpose by adapting contrastive learning and cross entropy respectively:

3.2.1. Contrastive Learning

The first proposed fine-tuning strategy is through adapting contrastive learning. Even though contrastive learning has been primarily used for fine-tuning for tasks that require pairwise textual understanding, it is possible to reformulate it to generate a training signal for a similarity-based classification task. The proposed contrastive learning based fine-tuning algorithm is outlined in Algorithm 3. Firstly, for every input-label pair, a new contrastive label "1" is assigned, representing them as similar pairs. This is followed by randomly pairing inputs with a class they do not belong to and assigning contrastive label "0" representing dissimilarity, akin to the negative samples generation method used in [33]. Finally during training, each input pair along with its contrastive label is used to calculate the contrastive loss according to $Loss_{con}$ defined in Eq. (2), where $\mathtt{dis}(\mathtt{sim}[] = 1 - \mathtt{sim}[])$, n, m are the indices of the text example and class description respectively, and d is the margin which defines the tolerated distance between similar and dissimilar inputs. Fig. 5 shows an illustration of contrastive learning for both the Bi-Encoder and Cross Encoder classifiers for input pair $T[n]$

Algorithm 3: SBERT fine-tuning with contrastive training objective for single-text classification

```
Input: $C$: List of Descriptive Class Definitions,
$T$: List of Training Texts,
$L$: List of Training Text Labels,
$n_{tr}$: Ratio of Negative Training Examples
Output: $\mathrm{SBERT}_{con}$: Contrastively fine-tuned SBERT classifier
$c = |C|$;
$t = |T|$;
// during training examples preparation
$examples = \emptyset$ ;
for $i = 1$ to $t$ do
    // Add actual examples
    $examples \leftarrow examples \cup [\{"text" : (T[i], C[L[i]]), "label" : 1\}]$;
end
for $j = 1$ to $t \times n_{tr}$ do
    // Randomly pair an input with another class
    $T_{\mathrm{rand}} = \{T[n] \mid 1 \leq n \leq t\}$;
    $C_{\mathrm{neg}} = \{C[m] \mid m \neq L[n], 1 \leq m \leq c\}$;
    $examples \leftarrow examples \cup [\{"text" : (T_{rand}, C_{neg}), "label" : 0\}]$;
end
// during training loss calculation
for $batch$ in $training\_batches$ do
    $loss = 0$ ;
    for $example$ in $batch$ do
        if Bi-Encoder then
            $T_{emb}, C_{emb} \leftarrow \mathrm{SBERT}_{con}(example["text"])$;
            $sim \leftarrow \mathtt{cosim}(T_{emb}, C_{emb})$;
        end
        if Cross Encoder then
            $sim \leftarrow \mathrm{SBERT}_{con}(example["text"])$;
        end
        $loss \leftarrow loss + \mathrm{Loss}_{contrastive}(sim, example["label"])$
    end
    $\mathrm{SBERT}_{con} \leftarrow \mathtt{train}(loss)$
end
return $\mathrm{SBERT}_{con}$
```

Algorithm 4: SBERT Fine-tuning with Cross Entropy training objective

```
Input: $C$: List of Descriptive Class Definitions,
$T$: List of Training Texts,
$L$: List of Training Text Labels
Output: $\mathrm{SBERT}_{crs\_entr}$: SBERT fine-tuned with Cross Entropy
$c = |C|$;
$t = |T|$;
// during training examples preparation
$examples = \emptyset$;
for $i = 1$ to $t$ do
    $examples \leftarrow examples \cup [\{"text" : T[i], "label" : L_{crs\_entr}(L[i])\}]$;
end
// during training loss calculation
$loss = 0$ ;
for $example$ in $training\_batch$ do
    if Bi-Encoder then
        $T_{emb}, C[1:c]_{emb} \leftarrow \mathrm{SBERT}_{crs\_entr}(example["text"], C[1:c])$;
        $sims[1:c] \leftarrow \mathtt{cosim}(T_{emb}, C[1:c]_{emb})$;
    end
    if Cross Encoder then
        $sims[1:c] \leftarrow \mathrm{SBERT}_{crs\_entr}(example["text"], C[1:c])$;
    end
    $\hat{sims} \leftarrow \mathtt{softmax}(sims)$;
    $loss \leftarrow loss + \mathrm{Loss}_{cross\_entropy}(\hat{sims}, example["label"])$
end
$\mathrm{SBERT}_{crs\_entr} \leftarrow \mathtt{train}(loss)$
return $\mathrm{SBERT}_{crs\_entr}$
```

and $C[m]$ corresponding to the nth text input example and the definition of the mth with their contrastive label L_{con} having value of either 0 or 1.

$$Loss_{Con} = L_{con}\mathtt{dis}(sim[n,m]) + (1 - L_{con})\mathtt{max}(0, d - \mathtt{dis}(sim[n,m])) \qquad (2)$$

3.2.2. Cross Entropy

The second proposed fine-tuning strategy is through adapting cross entropy. Cross entropy is commonly used in training standard models for classification tasks. However, it is only used as a loss function during contrastive learning in *SBERT* for tasks that require paired inputs like entailment, as opposed to a single-text classification task. It is possible to implement cross entropy in the reformulated similarity-based text classifiers by replacing class logits from traditional classifiers with similarity values for each class. Detailed steps for the fine-tuning process is outlined in Algorithm 4. Firstly, training examples consisting of an input text $T[i]$ and the corresponding ground truth class Label $L[i]$ are prepared, with $L[i]$ being transformed into a one-hot encoded vector $L_{crs\_entr}(L[i])$ where all elements are 0 except for the element corresponding to $L[i]$ which is 1. During training, similarities $sims$ between the input text and every class C are computed, followed by a $softmax$ function (Eq. (3)) which converts all elements in $sims$ into values bounded between 0 and 1, creating $\hat{sims}$, which allows the cross entropy loss function $Loss_{cross_{e}ntropy}$ (Eq. (4)) to be applied (Cross entropy is designed to be applied to probabilities). Fig. 6 illustrates the cross entropy loss calculation process for both the Bi-Encoder and Cross Encoder classifiers for text input example $T[n]$, class descriptions $C[1 : c]$, and ground truth class $L[n]$.

$$softmax(sim[i]) = \frac{e^{sim[i]}}{\sum_{j=1}^{c} e^{sim[j]}} = \hat{sim}[i] \qquad (3)$$

$$Loss_{cross\_entropy} = -\sum_{i=1}^{c} L_{crs\_entr}[i]\log(\hat{sim}[i]) \qquad (4)$$

4. Case study

To fulfill the research questions set out, a case study was carried out to investigate the use of similarity-based text classification in analyzing road maintenance texts. Section 4.1 discusses the dataset used in this case study, along with data processing steps. Section 4.2 elaborates on the task simulated in this case study. Section 4.3 outlines representative state of the art techniques that are used as comparison baselines. Section 4.4 discusses the implementation details of the experiments carried out. Finally, Section 4.5 explains all metrics quoted in this paper, including their interpretation and significance.

4.1. Dataset

The dataset used in this case study is called *"Street Construction Inspections and Corrective Action Requests"*, available from the NYC Open Data [47] which is a recommended TLP resource [6]. It is a collection of inspection comments written by Department of Transport inspectors where each entry records an instance of a road code violation of third party roadwork performed on public roads. They inspect for violations of the conditions set out in licenses to minimize disruptions and safety risks to road users. Each comment is classified to a road code that is being violated as set out by the Street Works Manual [48], with each code having a descriptive textual description of what constitutes a violation. A few pairs of inspection comment and their associated

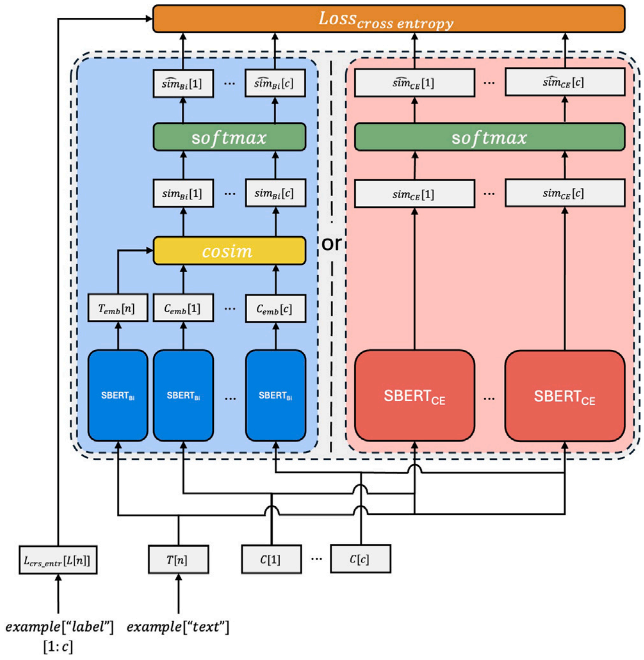

$Loss_{cross\ entropy}$
$\hat{sim}_{Bi}[1]$ ··· $\hat{sim}_{Bi}[c]$
softmax
$sim_{Bi}[1]$ ··· $sim_{Bi}[c]$
cosim
$T_{emb}[n]$
$C_{emb}[1]$ ··· $C_{emb}[c]$
SBERT$_{Bi}$
or
$\hat{sim}_{CE}[1]$ ··· $\hat{sim}_{CE}[c]$
softmax
$sim_{CE}[1]$ ··· $sim_{CE}[c]$
SBERT$_{CE}$ ··· SBERT$_{CE}$
$L_{crs\_entr}[L[n]]$
$T[n]$
$C[1]$ ··· $C[c]$
example["label"] [1: c]
example["text"]

Fig. 6. Illustration of the proposed cross entropy based fine-tuning process for the nth test input $T[n]$ and class definitions $C[1:c]$ for both the Bi-Encoder (left) and Cross Encoder (right) configurations.

violated road code definitions are listed in Table 1. In the dataset, the inspection comments T are under the column *"DetailsofViolation"* and the associated road code definitions L under *"NOVCodeDescription"*. There is heavy use of technical language in the free text comments, making it a good representation of domain-specific textual data in road maintenance and wider AEC fields.

4.1.1. Data cleaning

The dataset was cleaned in preparation for the case study. Firstly, it was found that there were road code duplications from overlapping definitions and character encoding errors, and the corresponding classes were combined to ensure consistency. Table 2 outlines examples of duplicate classes along with explanations as to why they are considered as such. Secondly, road codes that were violated fewer than 100 times are excluded from the final dataset as there needs to be sufficient evaluation examples per codes to allow for representative conclusions to be drawn.

The final dataset after class consolidation and filtering contains 408,080 entries, corresponding to 83 road code classes. The average number of words in L is 11 while that of T is 39. The large volume of data allows for reliable evaluation of the proposed classifiers and more representative conclusions to be drawn. The large number of road code classes (more than most standard classification tasks where there are a few classes) resemble many road maintenance or AEC tasks where the complex nature of the application and the rigidness of rules and regulations create a wide range of possible categorizations.

4.1.2. Few-shot datset generation

As resource-efficiency is a key area of investigation for this work, model training was performed on small subsets of the full cleaned dataset. Entries were randomly sampled to form few-shot training subsets of between 1 and 10 examples per class. 10 was chosen as the maximum as it already has 830 total training examples (83 classes), which is approaching the boundary of most few-shot settings. Additionally, small validation subsets of roughly a third the size of the training sets were also randomly sampled, with each validation subset having at least 1 example per class. The rest of the dataset was used as testing data. As instability can occur with very small training datasets, for each few-shot scenario, 10 subsets were randomly sampled which were then trained and tested independently, similar to the process in [33]. The results are then averaged and the standard deviations are recorded.

Overall, the majority of the experiments were performed using the above described few-shot datasets (1, 2, 3, 4, 5, and 10-shot). However, additional focused edge case experimentation was also performed:

Most models were additionally trained using all available data to provide a point of reference for non-resource-constrained performance. This dataset uses 80% of examples as training, and 10% for validation and testing respectively.

For models that can perform under zero-shot settings, testing was performed using the full cleaned dataset as testing data.

Furthermore, a focused set of few-shot experiments were conducted on larger pre-trained models (~7B) to assess the impact model sizes have on performance. Considering their computationally expensive nature, the number of per-shot independent datasets was set to three and the total per-shot testing examples was selected as 15,000.

4.2. Task formulation

The task is formulated as a single-label, many-classes text classification task, where road inspection comments are classified according to one of 83 pre-defined categories through the use of descriptive class definitions. Specifically, this case study simulates the task of classifying roadwork inspection comments T according to the descriptive road code violation definitions C. A similarity-based classification criteria is used where for the nth comment $T[n]$, the code with the highest semantic similarity is selected as the prediction $P[n]$. Fig. 7 shows the simulated pipeline of classifying textual comments of road code violations resulting from roadwork inspections into violated codes, and outlines the classification process of both the Bi-Encoder Only Classifier and the Reranked Cross Encoder Classifier from Sections 3.1.1 and 3.1.2 for roadwork inspection comments, with $P_{Bi}[n]$ and $P_{CE}[n]$ being the respective outputs. The predictions are then compared with the ground truth classification label $L[n]$.

4.3. Baselines

In order to answer the research questions, it is important to evaluate the proposed classifier's performance and resource-efficiency against alternative, state of the art methods. The following techniques are used as comparison baselines for the task of classifying roadwork inspection comments into violated road codes:

1. **Traditionally Fine-tuned Text Classifiers**: Traditionally fine-tuned text classifiers utilize the architecture illustrated in Fig. 2. Only the main classifier parameters are pre-trained and class-specific logits are produced using a task-dependent and uninitialized classification head. This is the most common text classifier configuration with transformers, but it is not designed for low-resource model adaptation. This baseline enables the comparison between the proposed classifier and the most common conventional architecture.
2. **Few-Shot Learning (SetFit)**: SetFit is a popular few-shot learning text classifier that also leverages the *SBERT* architecture (Section 2.4.1). Works such as [37] also included this baseline due to it being one of the most representative approaches in few-shot text classification. The biggest differences between SetFit and the proposed similarity-based classifier are that similarity is not a classification criterion in the former, and that during training, similarity is only calculated between inputs and not against any form of static class definitions. This baseline enables the comparison between the proposed classifier and a state of the art few-shot learning technique that does not require prompting. Note that SetFit was not tested on the all-data setting as its implementation requires the generation of pairs between every single input, which would be unscalable with more than 300,000 training examples in that scenario.

Table 1
Examples of free text inspection comments T along with the descriptive definitions of their associated violated road codes L.

Free text inspection comments T	Definitions of road code violations L
I OBSERVED DEBRIS (COLD PATCH) BELONGING TO THE ABOVE RESPONDENT OBSTRUCTING ROADWAY GUTTERS. SITE UN ATTENDED. LOCATED OPPOSITE 265 23RD ST. If not admitting the charge, you MUST APPEAR IN PERSON.	*Debris/construction materials obstructing gutters/sidewalk, etc.*
Bubbled pedestrian warning devise not install on ped ramp as required by DOT specs.	*Except as in NYC Administrative Code 19-152, failure to install pedestrian ramp as per DOT drawings*
A/T/P/O I observed unsecured Verizon telephone cable hanging (from Verizon pole section) on wooden pole65359 located on SE Corner of intersection in violation of 34 RCNY 2-20(j)(4). CAR#20226282105 was issued on 12/2/2022. If not admitting the charge, you MUST APPEAR IN PERSON.	*Failure to repair non-city electrical and/or non-electrical equipment within the required time frame.*

Table 2
Examples of duplicate codes, each row representing a duplicate.

Code 1	Code 2	Reason for duplication
Commercial refuse container/debris obstructing sidewalks, gutters, crosswalks or driveway	*Commercial refuse container/debris obstructing sidewalks, gutters, crosswalks, bike lanes or driveway*	**Semantically identical**: No clear case when 1 would be preferable to 2
Failure to post âSteel Plates Aheadâ or âRaise Plowâ sign; failure to countersink plates flush with roadway	*Failure to post "Steel Plates Ahead" or "Raise Plow" sign; failure to countersink plates flush with roadway*	**Character encoding error**: The "â" is a result of a character encoding error originating from the publisher

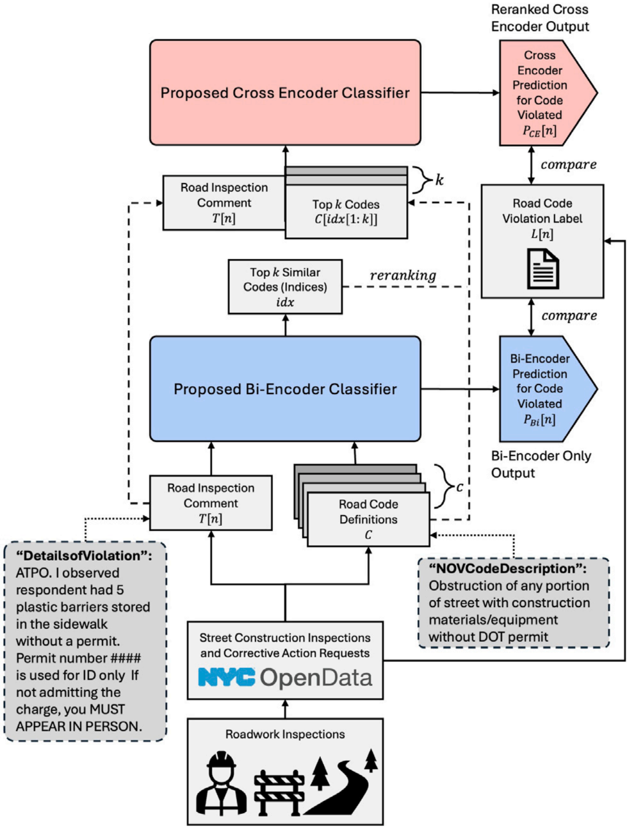

Reranked Cross Encoder Output
Proposed Cross Encoder Classifier
Cross Encoder Prediction for Code Violated $P_{CE}[n]$
k
compare
Road Inspection Comment $T[n]$
Top k Codes $C[idx[1:k]]$
Road Code Violation Label $L[n]$
Top k Similar Codes (Indices) idx
reranking
compare
Proposed Bi-Encoder Classifier
Bi-Encoder Prediction for Code Violated $P_{Bi}[n]$
Bi-Encoder Only Output
Road Inspection Comment $T[n]$
Road Code Definitions C
c
"DetailsofViolation": ATPO. I observed respondent had 5 plastic barriers stored in the sidewalk without a permit. Permit number #### is used for ID only If not admitting the charge, you MUST APPEAR IN PERSON.
"NOVCodeDescription": Obstruction of any portion of street with construction materials/equipment without DOT permit
Street Construction Inspections and Corrective Action Requests
NYC OpenData
Roadwork Inspections

Fig. 7. Illustration of the classification process for the nth roadwork inspection comment $T[n]$ according road code definitions C. $P_{Bi}[n]$ and $P_{CE}[n]$ refer to the code violation prediction of the Bi-Encoder Only and Reranked Cross Encoder Classifiers respectively.

4.4. Implementation details

To implement the proposed and baseline classifiers, various Python libraries were utilized, including the Transformers library and Sentence Transformer library. Models were trained using an RTX 4080 with 16 GB of VRAM. Codes used are available online.[1]

Table 3
Bi-Encoder training hyperparameters used. Search range in square brackets.

Hyperparameters	Fine-tuning	
	Contrastive	Cross Entropy
Learning rate	$2e{-}5_{[1e-7,1e-4]}$	$2e{-}5_{[1e-7,1e-4]}$
Batch size	$16_{[4,32]}$	$16_{[4,32]}$
Scale	N/A	$20_{[1,60]}$
Margin	$0.7_{[0.5,1.5]}$	N/A
Neg. ratio	$16_{[0,82]}$	N/A

For each baseline and proposed classifier, pre-trained models were selected based on benchmarks and validation performances. In addition, the validation sets were also used to determine hyperparameters for each approach. All models were trained to 20 epochs, with the best model being selected automatically using accuracy scores on the validation sets. Additionally, in the Reranked Cross Encoder Classifier, the Bi-Encoder and Cross Encoders were trained using the same dataset though through independent training processes. Details for each classifier are included below:

4.4.1. Proposed: Bi-Encoder

For the Bi-Encoder model in both architectures, *all-mpnet-base-v2* with 109 Million was chosen as the pre-trained model as recommended by the official *SBERT* documentation [49]. It also performed better than alternative models such as *stsb-mpnet-base-v2* and *stsb-roberta-base-v2* which were trained on semantic textual similarity tasks. The contrastive learning approach has additional hyperparameters *margin* and *negative ratio*, representing d in Eq. (2) and n_{tr} in Algorithm 3 respectively. The cross entropy approach includes a scaling parameter for the overall loss. Table 3 lists out the hyperparameter values used.

4.4.2. Proposed: Cross Encoder

The Cross Encoder used in the Reranked Cross Encoder Classifier used *quora-roberta-base* as the pre-trained model, as it performed best compared to other pre-trained models tested (*bert-base-uncased, roberta-base* and *stsb-roberta-base*) and is also listed as a recommended model in the official documentation. As this architecture utilizes the output of a Bi-Encoder model, the classifier is trained using the same dataset. Therefore, for contrastive learning, the number of negative training examples is the same as the Bi-Encoder model. Additionally, when training the Cross Encoders, a subsampling approach is taken where only a subset of classes are included in each cross entropy calculation, though the ground truth will always be included. This is to reduce the computational load due to the need to encode every unique pair for training. The hyperparameters used for the two fine-tuning approaches are outlined in Table 4.

[1] https://github.com/Soppy-Minotaur/Sim_Fit.git.

Table 4
Cross Encoder training hyperparameters used. Search range in square brackets.

Hyperparameters	Fine-tuning	
	Contrastive	Cross Entropy
Learning rate	2e−5$_{[1e-7,1e-4]}$	2e−5$_{[1e-7,1e-4]}$
Batch size	16$_{[4,32]}$	16$_{[4,32]}$
Scale	N/A	20$_{[1,60]}$
Margin	0.7$_{[0.5,1.5]}$	N/A
Neg. ratio	N/A	10$_{[3,20]}$

Table 5
SetFit training hyperparameters used. In both the embedding and classifier fine-tuning phases, the first and second values are for the SBERT encoder and classification head respectively. As the classification head is not present during the Embedding phase, its values are represented as “N/A”.

Hyperparameters	Fine-tuning	
	Embedding	Classifier
Learning rate	2e−5, N/A	1e−5, 1e−2
Batch size	16, N/A	16, 2
No. epochs	1, N/A	1, 16

Table 6
∼100M pre-trained models used and their associated training hyperparameters (search range in square brackets).

Pre-trained model	Hyperparameters		
	No. parameters	Learning rate	Batch size
bert-base-uncased	110 Million	4e−5$_{[1e-6,1e-4]}$	16$_{[4,32]}$
roberta-base	125 Million	2e−5$_{[1e-6,1e-4]}$	16$_{[4,32]}$
xlnet-base-cased	110 Million	2e−5$_{[1e-6,1e-4]}$	16$_{[4,32]}$

Table 7
∼7B pre-trained models used and their associated training hyperparameters (search range in square brackets).

Pre-trained model	Hyperparameters		
	No. parameters	Learning rate	Batch size
Llama-2-7b	6.74 Billion	2e−4$_{[1e-3,1e-5]}$	3
Mistral-7B-v0.1	7.24 Billion	2e−4$_{[1e-3,1e-5]}$	3
Meta-Llama-3-8B	8.03 Billion	2e−4$_{[1e-3,1e-5]}$	3

4.4.3. Baseline: SetFit

SetFit models were trained using *paraphrase-multilingual-mpnet-base-v2*, which is the same pre-trained *SBERT* model as the one used in the original paper [33]. The hyperparameters used also followed the paper and recommendations from the official library, listed in Table 5. They include values for both the body (*SBERT*) and head (classification head) at the two training stages.

4.4.4. Baseline: Traditionally fine-tuned classifiers

Three ∼100M pre-trained transformer models listed in Table 6 were used as the primary non-few-shot baselines as they are commonly used for sequence classification. Notably in a comparison study [50], they were the three best performing pre-trained models with roughly equal numbers of parameters to models used in the proposed similarity-based classifiers and SetFit. By having similar sizes, it ensures that any difference in performance is due to the classification algorithms instead of the number of model parameters.

However, in order to validate the absolute performance of the proposed classifier compared to the state of the art, additional focused experimentation was performed with three much larger baseline pre-trained models with ∼7B parameters (see Table 7). They have roughly 70 times the number of parameters of other pre-trained models tested in this work and are widely cited in other literature due to their high performance [37,38]. Note that due to the high computational costs of these bigger models, for every number of shots tested, only three independent subsets were employed along with a total testing set of 15,000 texts.

4.5. Metrics

The following metrics are used to evaluate the performances of the proposed and baseline classifiers. They are in line with common metrics reported for text classification tasks and measure the relative rate of appearance of true positive (tp), true negative (tn), false positive (fp), and false negative (fn) according to the following equations:

$$accuracy = \frac{tp + tn}{tp + tn + fp + fn} \tag{5}$$

$$f1 = \frac{tp}{tp + \frac{1}{2}(fp + fn)} \tag{6}$$

Higher values are desired for all metrics. Averaging across classes is done to output an overall metric value. For this work, $accuracy$ values reported are micro-averaged (averaged across test examples) and $f1$ values are macro-averaged (averaged across classes). Micro-averaged metrics indicate overall performances but can be skewed by class imbalance where classes have significantly different number of testing examples. Macro-averaging assign equal weights to classwise metrics, allowing for performances of less frequent classes to be captured. It is worth noting that micro-averaged $accuracy$ values are equivalent to micro-averaged $f1$ scores in single-label tasks, such as this case study.

Overall metric results quoted are mean values across the ten independent datasets for each test setting as detailed in Section 4.1, except for experiments with larger ∼7B models which were averaged over three. Note that the standard deviations listed account for the stochastic nature of both the sampling process during training and during few-shot dataset generation, which differs from most works where it only considers the former. This inflates the variance as each individual training session utilizes a different set of subsampled training data which would create models according to independent distributions.

5. Results and discussions

In order to answer the research questions, the performances of all proposed models and baseline models are analyzed. Section 5.1 evaluates all proposed models to determine the optimal similarity-based classifier (*RQ2* and *RQ3*). Section 5.2 provides insight into how the proposed classifiers function, justifies the use of descriptive class definitions for classification (*RQ1*), and visualizes the effects of fine-tuning (*RQ3*). Section 5.3 uses the optimal proposed classifier as determined above to compare against various baseline models which are designed for resource-constrained (few-shot) or performance-focused (traditionally fine-tuned) applications, allowing the proposal to be validated against state of the art and widely popular techniques and quantify its suitability in acting as a text classifier in road maintenance and the wider AEC context (*RQ4*). Finally, Section 6 further explores and consolidates important findings in the case study.

For completeness, in addition to diagrams and tables in the following sections, Table 14 in Appendix records averaged $f1$ and $accuracy$ values (standard deviation in subscripts) for all ∼100 million parameters proposed and baseline models along with SetFit.

5.1. Performance comparison between proposed models

The performances of the different proposed classifiers are first analyzed to determine both the optimal architecture and fine-tuning regime, referencing *RQ2* and *RQ3* respectively. There are two architectures and training methods proposed: Both the Bi-Encoder Only and Reranked Cross Encoder configurations were experimented with Contrastive Learning and Cross Entropy training, resulting in 4 sets of data.

Fig. 8 plots the $f1$ and $accuracy$ values for all proposed models trained with different numbers of training examples per class, along with results from the best performing baseline few-shot model (SetFit) for reference. Note that all proposed approaches perform better than the baselines in most few-shot settings, with more detailed analysis later in Section 5.3.

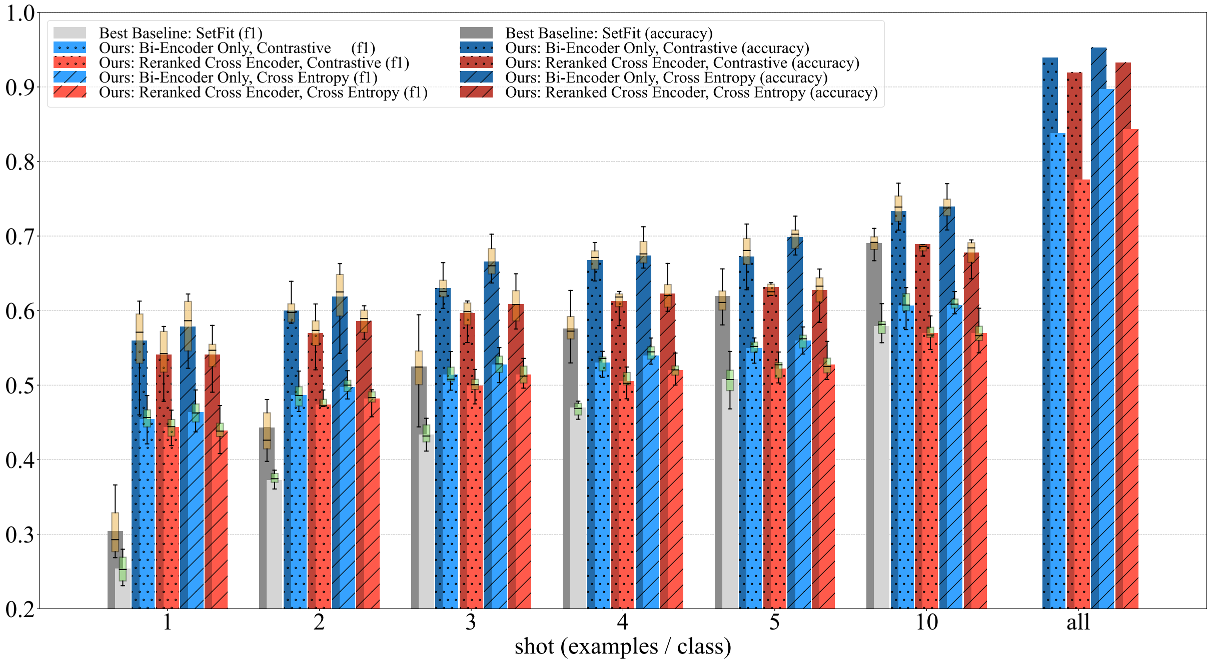


Fig. 8. Results in all training scenarios for all proposed models along with the best performing baseline (SetFit) for reference. Box plots are overlaid to summarize the results of the ten individual datasets per shot. Exact values and standard deviations are included in Table 14. Front bars represent $f1$ scores, rear bars represent *accuracy*. Blue bars represent the Bi-Encoder Only architecture, red bars represent the Reranked Cross Encoder architecture. Dotted bars represent models trained using contrastive learning, while striped bars represent models trained using cross entropy. All proposed models exhibit comparable results, though the Bi-Encoder Only model trained with cross entropy consistently demonstrates a lead.

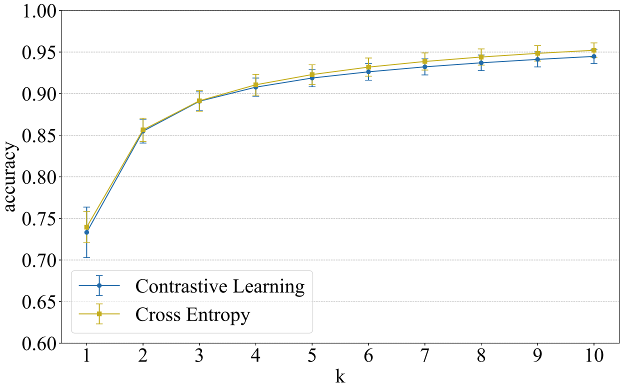


Fig. 9. Probability of the ground truth being in the top k most similar classes for both Contrastively and Cross Entropy trained Bi-Encoder models.

5.1.1. Determining optimal k value

In order to implement the Reranked Cross Ecnoder approach, the architecture-specific parameter k correlating to the number of candidates transferred from the Bi-Encoder to the Cross Encoder needs to first be determined. This decision is based on the trade-off between accuracy and number of encoding operations, where higher k values increase accuracy at the expense of a higher computational load. Fig. 9 plots the probability that the top k results contain the ground truth label, which can be seen as the theoretical maximum accuracy attained at each k value. 4 was selected for this parameter as it achieves a theoretical maximum accuracy of more than 90% while still limiting the computational costs.

5.1.2. The Bi-Encoder only configuration is a better architecture

To answer *RQ2*, the difference in performance between the Bi-Encoder Only and Reranked Cross Encoder architectures is first analyzed. From Fig. 8, the main observation is that the Bi-Encoder Only models perform better than the Reranked Cross Ecnoder models. The choice between them were initially hypothesized to be a trade-off between speed and performance. However in few-shot settings this does not seem to be the case. As shown in Table 8, for every few-shot and all-data testing scenario, the Bi-Encoder Only models performed better, with an average few-shot $f1$ lead of 4% and 4.4% for contrastively trained and cross entropy trained models respectively. The gap is widened when the full training dataset becomes available, with the lead of the Bi-Encoder Only model being 6.7% and 10.2% for the two training approaches. As this pattern is consistent with both micro-averaged *accuracy* and macro-averaged $f1$ values, it means that this lead applies to both overall performance and those of individual classes. Moreover, as the above observations remain consistent with the two training approaches proposed, it indicates that training strategies are independent of the relative performance between the two classifier architectures. They show that the Bi-Encoder Only approach exhibit generally better performances than the Reranked Cross Encoder approach, and that it is more able to correctly associate road inspection comments with road codes.

Table 8
Changes in $f1$ scores from Bi-Encoder Only to Reranked Cross Encoder configurations for contrastively trained (Con.) and cross entropy trained (C. E.) models. (+ means Reranked Cross Encoder is better).

Shots	Few-shot						All
	1	2	3	4	5	10	
Con.	−2.4%	−2.7%	−2.8%	−4.7%	−5.1%	−6.1%	−7.5%
C. E.	−5.3%	−3.0%	−2.6%	−3.6%	−5.8%	−6.3%	−6.0%

The discrepancy in performance can potentially be attributed to both the choice of the pre-trained models and subsampling when training the Cross Encoders. Regarding the process for pre-trained model selection, the choice for the Bi-Encoders (*all-mpnet-base-v2*) is different from that for the Cross Encoders (*quora-roberta-base*) due to them having different architectures. These pre-trained models were themselves built from different base models and fine-tuned under different

Table 9
Changes in $f1$ scores from training with Contrastive Learning to Cross Entropy for the Bi-Encoder Only (Bi.) and Reranked Cross Encoder (R. C. E.) configurations. (+ means Cross Entropy is better).

Shots	Few-shot						All
	1	2	3	4	5	10	
Bi.	+2.0%	+2.1%	+2.7%	+1.8%	+1.8%	+0.1%	+7.1%
R. C. E.	−1.0%	+1.8%	+2.9%	+2.9%	+1.0%	−0.0%	+3.1%

datasets. For instance, *all-mpnet-base-v2* is based on *mpnet-base* and pre-trained using a variety of datasets including Reddit comments [51] and citation pairs [52]. On the other hand, *quora-roberta-base* is based on *roberta-base* and further pre-trained on the Quora Duplicates Questions dataset. The difference in the model design and preparation processes contribute to their difference in performance. The significance of this factor is further examined in Section 6.1. On the other hand, the use of subsampling when training the Cross Encoder is necessary when Cross Entropy is used as it reduces the number of encoding operations needed. This can potentially impact the strength of the training signal produced as only 10 out of 83 classes are represented when training, with 10 having been chosen as a hyperparameter during initial testing as a balance between speed and performance. However, as this operation is only performed for cross entropy-trained models, this still does not explain the performance discrepancy in contrastively trained models that exhibit the same patterns. It is expected that this factor only contributes to a small extent to the observation.

It is important to note that the differences in performance between the two proposed architectures are small. All proposed models performed better than the best performing baseline (SetFit) in most few-shot settings. The lead is especially big during lower-shot testing, with the scores almost doubling that of the baseline at 1-shot. This demonstrates that at very resource-constrained settings, both architectures can produce better quality classification outputs than even specialized few-shot techniques. Therefore, both configurations are better suited for classifying domain-specific road maintenance texts than state-of-the art existing approaches when there is insufficient training data. More comparison with the baseline models are included in Section 5.3.

The case for using the Bi-Encoder Only approach in an inspection text automation pipeline is not limited to performance. In practical deployment for AEC operations like road maintenance, inference speed is also relevant. As mentioned before, the Cross Encoder requires encoding every input-class definition pair, a characteristic that does not scale well when there are many classes, as is the case with many AEC applications due to the tightly regulated, sensitive, and complex nature of many domains. For instance in this case study on classifying road inspection texts where there are 83 road codes that can be violated, the Cross Encoder requires 40 min to perform inference compared to 5 min for the Bi-Encoder. The difference in speed will be even greater when there are more testing examples and classes. By limiting the computational power required to perform automated text classification, fewer computing resources will be required which can contribute to improving the scalability of deployment and the rate of information output.

Overall, the Bi-Encoder Only classifier is both more performant and faster at inference, making it better than the Reranked Cross Encoder configuration. It is able to classify road inspection texts both more accurately and faster, both of which are important factors of consideration in road maintenance and wider AEC contexts. This answers *RQ2* in that the Bi-Encoder Only classifier is the most optimal structure for the similarity-based classifier for use in an automated road maintenance (and wider AEC) text classification pipeline.

5.1.3. Cross Entropy trains better models

With reference to *RQ3*, as fine-tuning SBERT for similarity-based single-text classification has not been explored, the two proposed fine-tuning strategies using the Contrastive Learning and Cross Entropy approaches are evaluated. This can provide evidence as to which model adaptation technique is optimal for a similarity-based classifier in resource-constrained engineering applications.

As seen on Fig. 8, models trained with the Cross Entropy loss (represented by striped bars) perform better than those trained using Contrastive Learning. This applies to all levels of data-constraints experimented with for the Bi-Encoder Only classifiers. The percentage improvements of models trained with Cross Entropy over those trained with Contrastive Learning are shown in Table 9, where the average few-shot lead for the Bi-Encoder Only and Reranked Cross Encoder architectures are 1.8% and 1.3% respectively. Interestingly, the lead when all data is available becomes significantly greater at 7.1% and 3.1% respectively, indicating that the Cross Entropy Loss is better able to capitalize on having abundant training examples in tuning model parameters.

As this trend is consistent across the two architectures proposed, it is independent of how the classifiers are structured. This observation is expected as Cross Entropy is commonly used for single-text classification tasks. By adapting it to similarity outputs of *SBERT*, it can effectively mimic standard classifiers by replacing logits with similarity scores. On the other hand, even though contrastive learning is commonly used to train *SBERT*, it is mostly used to train for paired classification task, or in the case of SetFit used only to train an embedding model and not a classifier. The training signal it generates is not as effective as that of Cross Entropy as it can only focus on a pair of similar and dissimilar inputs at a time, making it more difficult for the model to capture the wider task context.

Therefore, it can be inferred that the Cross Entropy training approach inspired by standard classifier fine-tuning can produce models that are more effective at classifying road maintenance texts than the contrastive approach inspired by standard SBERT pairwise fine-tuning. This answers *RQ3*.

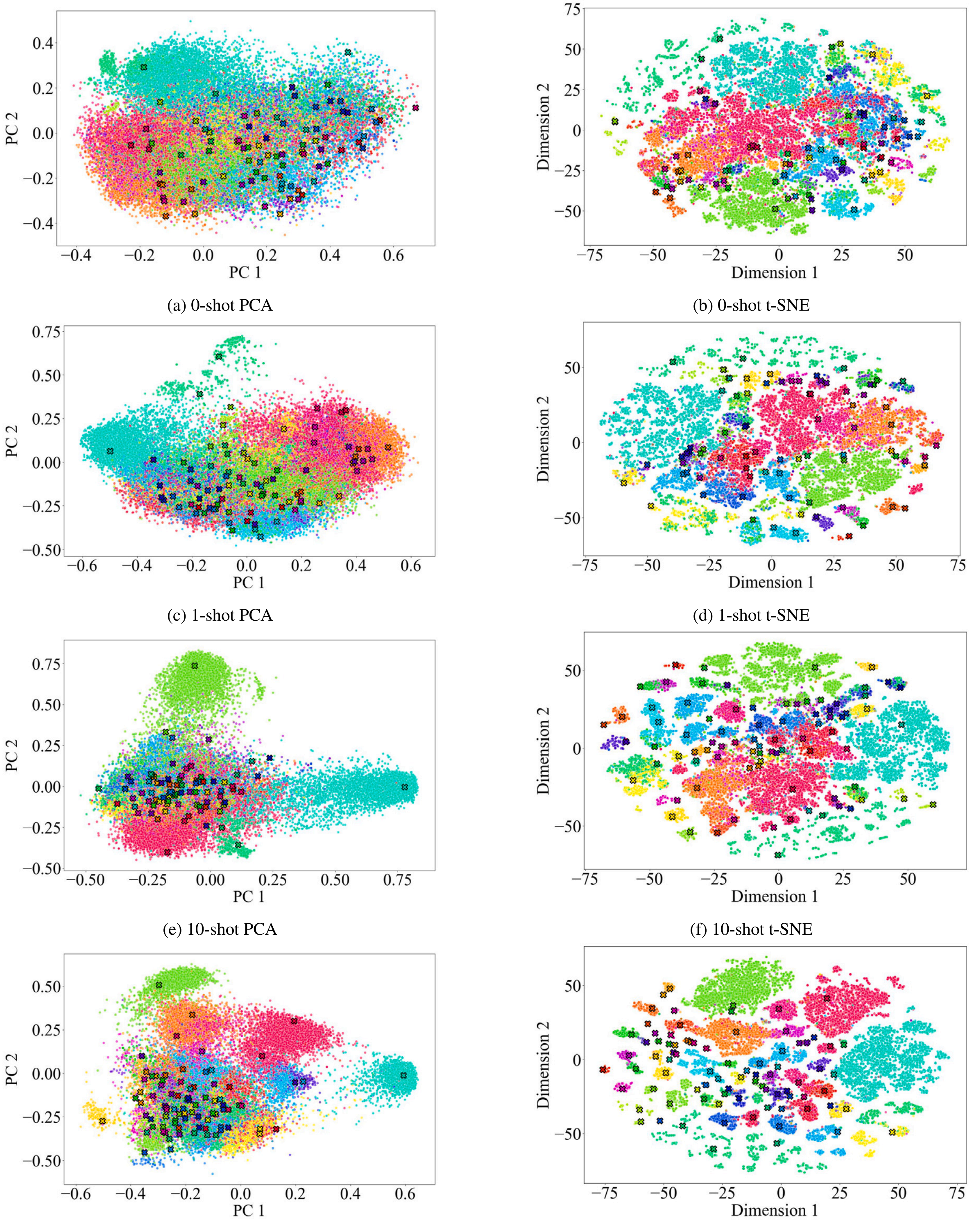
5.2. Understanding how the proposed classifiers function

As discussed in Section 3, the proposed classifier is distinct from other works in that it directly utilizes semantic similarity with descriptive class definitions as a classification criterion and that it involves fine-tuning *SBERT* for a single-text classification task. This section aims at determining whether the proposed classification criterion is valid (*RQ1*) and further quantifies the importance of fine-tuning even with limited data (*RQ3*). This can provide insight into how the proposed classifier functions.

5.2.1. Visualization of the embedding space

Bi-Encoders produce semantically-informative embeddings as an intermediate step in computing similarity scores. Visualizing the embedding space can provide insight into how accurately the models differentiate between classes and associate class examples with descriptive class definitions. To produce 2D visualizations (from embedding dimension 768), Principal Component Analysis (PCA) and t-distributed Stochastic Neighbor Embedding (t-SNE) dimensional reduction were employed. Visualizations for different few-shot scenarios are plotted in Fig. 10, where each colored cluster represents embeddings of class examples, and crosses of the corresponding colors represent the embeddings of the description class definitions.

PCA visualizations are shown on the left column. PCA preserves data variance, meaning that two points physically close to each other are considered semantically similar. However, it is limited in its ability to create clear clusters. t-SNE visualizations are shown on the right column. t-SNE is a non-linear dimensional reduction technique which



(a) 0-shot PCA

(b) 0-shot t-SNE

(c) 1-shot PCA

(d) 1-shot t-SNE

(e) 10-shot PCA

(f) 10-shot t-SNE

(g) all-data PCA

(h) all-data t-SNE

Fig. 10. PCA (left column) and t-SNE (right column) plots for Bi-Encoders trained with different numbers of shots through Cross Entropy. Each color represents a single class and the associated crosses represent the embedding of the corresponding descriptive class definition. Clusters become better defined as training data increases, indicating that the proposed architecture and fine-tuning regimes can enhance the embedding capability of SBERT. Clusters roughly center around the descriptive class definitions, indicating their effectiveness as class representation.

is better at isolating clusters at the expense of distance distortion. Distances have no clear interpretable meanings, though clusters are more salient.

The following sections will analyze the visualization to answer both *RQ1* and *RQ3*.

5.2.2. Descriptive class definitions is an effective classification criterion

The validity of using descriptive class definitions as a classification criterion is first assessed to answer *RQ1*.

As the proposed classifier does not require a task-specific classification head, relying solely on cosine similarity with the descriptive

Table 10
Zero-shot performances for the proposed similarity-based classifier (Bi-Encoder Only) for different pre-trained models.

Pre-trained model	$f1$	$Accuracy$
all-mpnet-base-v2	0.254	0.265
stsb-mpnet-base-v2	0.249	0.255
stsb-roberta-base-v2	0.225	0.257

class definition as the classification criterion, there is no uninitialized parameters, meaning that it is able to be used without any fine-tuning in a zero-shot manner. Table 10 outlines the zero-shot performances of the Bi-Encoder Only configuration for different pre-trained *SBERT* models. The pre-trained models chosen for all other experiments in this case study (*all-mpnet-base-v2*) performed the best with an $f1$ of 0.254 and $accuracy$ of 0.265, though the lead is at most only around 10% against other pre-trained models. This is likely a result of slightly different pre-training objectives, where the $stsb$ models were both pre-trained with a semantic textual similarity objective. The $f1$ score of the zero-shot proposed classifier is actually the same as that of the best performing baseline (SetFit) with one example per class, meaning that in the absence of training data, just by leveraging the supplied class definitions as a classification criterion, it is able to output comparable results with a fine-tuned baseline model.

By observing the clusters in Fig. 10, it can be seen that in both dimensionality reduction techniques, each cluster is roughly centered around the descriptive class definitions (represented as crosses of the same color). Though this is already the case for some classes in the zero-shot setting, this pattern became clearer as the amount of training data increases. With just 1-shot, much more classes are pushed further towards the edges to form small groups.

Overall, the fact that an unadapted pre-trained model can correctly classify more than 26% of road inspection comments and that the embeddings of class examples and class definitions are associated mean that the assumption that road code definitions are semantically similar to the description of code violation instances is valid, and that it can be used effectively used as a classification criterion for road inspection comments, answering *RQ1*.

5.2.3. Effects of fine-tuning

As discussed in Section 2.4.3, similarity-based classification with *SBERT* has primarily been done in zero-shot settings, and that fine-tuning it for this purpose has not been explored. Therefore, to fully answer *RQ3*, it is important to understand how the proposed fine-tuning process affects the performance of the proposed similarity-based classifier.

It is hypothesized that the performance can be further improved by the addition of a small amount of training examples. In order to further understand how the classification criterion can work in a few-shot setting, results from zero and few-shot settings are compared. Fig. 11 shows the histogram of the classwise $f1$ scores for a zero-shot model overlaid on that of a 10-shot model. It illustrates how the number of classes with high $f1$ scores is significantly increased when given only 10 examples per class. For example, there were no classes with $f1$ of more than 0.75 in the untrained model, but around a third achieved better than that in the 10-shot model. In the mean time, the number of classes with very low scores are dramatically reduced, and only a small minority of classes exhibit this behavior in the 10-shot model. This shows that despite being able to be used theoretically without any training, fine-tuning the proposed classifiers can be very effective at rapidly adapting models to the domain context.

Further, by analyzing the progression of the clusters in Fig. 10, it can be seen that clusters become more well defined as more training data becomes available. As expected, the t-SNE plots show clearer separation between clusters at all settings due to its use of distance distortion. For t-SNE, there are many well separated classes even in few-shot settings, though it is clear that being able to access all training data can further enhance this. For PCA, most classes are cluttered in the middle in all training settings. However, at 10 examples per class, two clusters (cyan and green) manages to separate out from the center.

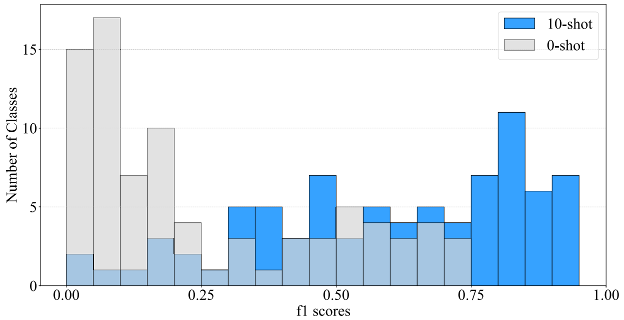


Fig. 11. Histogram of classwise $f1$ scores for the 10-shot Bi-Encoder Only model trained with Cross Entropy loss compared with an untrained 0-shot model. Fine-tuning significantly improved the performances of most classes as evident by the shift towards the right of the graph.

Overall, it was observed that upon fine-tuning there better well performing classes in terms of classwise metric values and that classes become more well separated in the embedding space. This indicates that fine-tuning using the proposed approach can boost the performance of the proposed classifier, further answering *RQ3*.

5.3. Performance comparison with baseline models

To understand how the proposed similarity-based classifier compares with alternative approaches and answer *RQ4*, its performance was evaluated against both state of the art traditionally fine-tuned and few-shot models. Fig. 12 shows the metric values for: the few-shot baseline in grey (SetFit), the traditionally fine-tuned baselines in green (*bert-base-uncased*, *roberta-base*, and *xlnet-base-cased*), and the best performing proposed classifier in yellow (Bi-Encoder Only fine-tuned with Cross Entropy). For completeness, Table 14 in Appendix outlines the $accuracy$ and $f1$ scores of all trained baseline models and proposed similarity-based classification models. All models used in this section have roughly 100 million parameters (except for SetFit which has roughly 300 million), allowing for results to be more comparable. Further experimentation a large model is included in Section 6.2.

5.3.1. Improvements over SetFit in few-shot settings

SetFit is one of the most representative and powerful few-shot text classifiers, specializing in tasks where there is limited data availability. Its few-shot capabilities are demonstrated in this case study where it surpassed traditional fine-tuning based methods that typically require more data to train. For example with just one training example, the few-shot SetFit classifier was able to attain an $f1$ score of 0.254, compared to much less than 0.1 for all traditional models. It maintained its lead over other baselines throughout all few-shot scenarios. This reaffirms the strength of SetFit compared to techniques that are not designed for data-constrained classification.

However, it can be observed from Table 14 that the proposed classifier has a considerable lead over even SetFit in few-shot settings. In a 1-shot setting, the lead is more than 80%, with the gap narrowing to only roughly 5% lead in a 10-shot setting. For the same level of performance in terms of $f1$ of around 0.46, the proposed classifier only requires one example per class whereas SetFit requires five. The performance discrepancy can be visualized by investigating the classwise metric values. Fig. 13 shows the number of classes that attained different $f1$ values for both SetFit and the proposed classifier in a 1-shot setting. As shown, SetFit results overlaid in grey exhibit generally worse performance where the majority of classes have very low $f1$ scores.

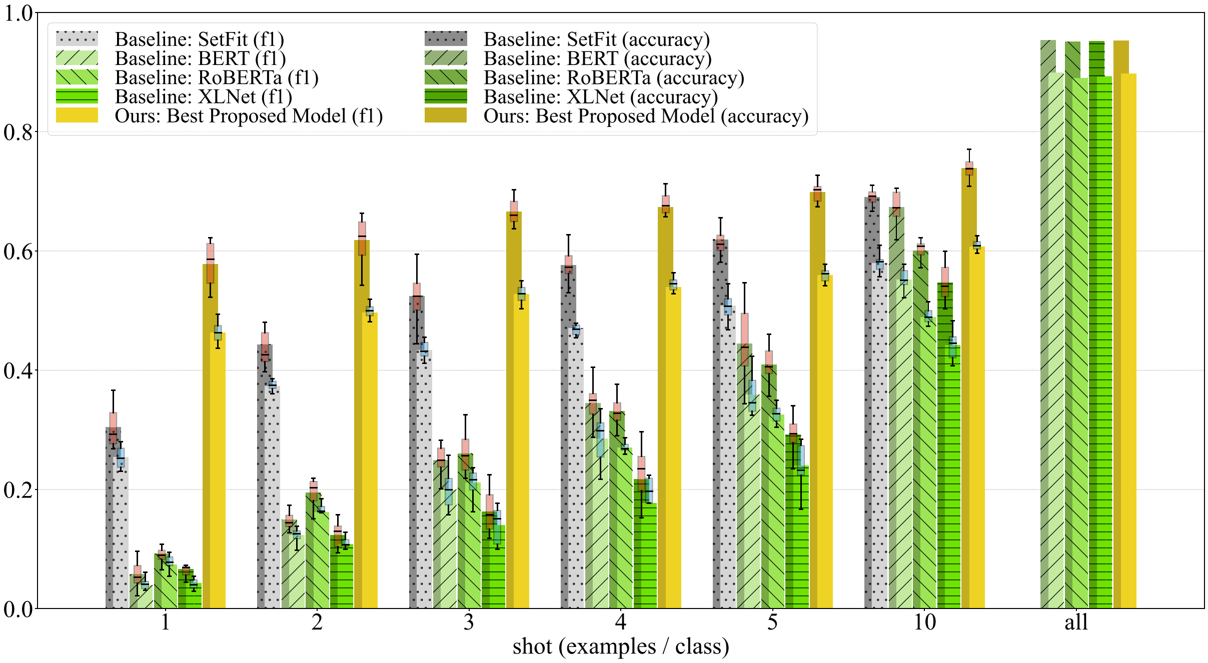


Fig. 12. Results in all training scenarios for ∼100M baseline models and the best performing proposed ∼100M model (Cross Entropy trained Bi-Encoder Only classifier). Box plots are overlaid to summarize the results of the ten individual datasets per shot. Exact values and standard deviations are included in Table 14. Front bars represent $f1$, rear bars represent *accuracy*. Grey bars represent the few-shot baseline SetFit, green bars represent traditionally fine-tuned baselines, and yellow bars represent the best proposed model. The best proposed model performs better than all baselines in all few-shot scenarios and performs comparably in the all-data scenario.

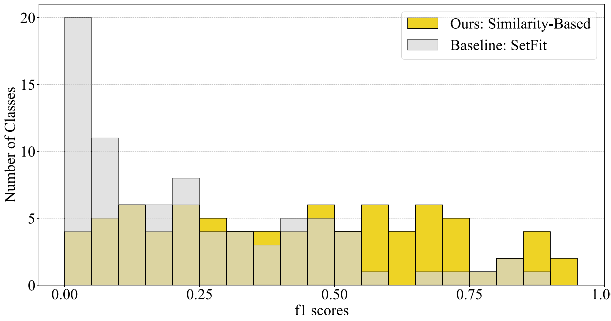


Fig. 13. Histogram of classwise $f1$ scores for the best performing proposed model overlaid with the SetFit model at a 1-shot setting. More classes exhibit higher scores for the proposed model as shown by the bars towards the right.

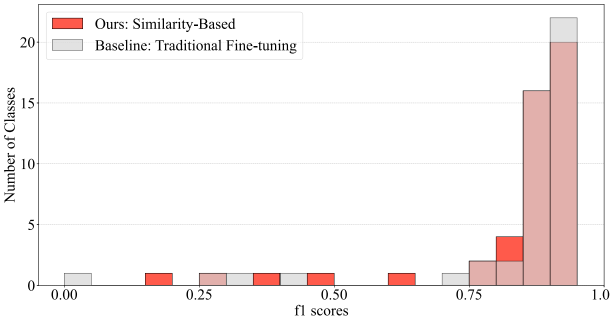


Fig. 14. Histogram of classwise $f1$ scores for the best performing proposed model overlaid with the best performing traditional fine-tuning technique in an all-data training scenario. They exhibit very comparable performance as seen by the almost complete overlap in classwise scores.

20 of the 83 classes occupy the lowest valued bin in the histogram. This is different from the results of the proposed classifier (in yellow) where even though few achieve very high $f1$ values, most still achieve moderate values. The are only 4 classes occupying the lowest valued bin. This shows that both in terms of overall metric values and the performance of individual classes, the proposed classifier is superior.

The difference in performance is a result of effective use of additional task information supplied by the descriptive definition of road codes which are directly used as the classification criterion. Despite both being *SBERT*-based classification approaches, the proposed classifier does not solely rely on training examples to acquire task information, instead using its ability to calculate semantic similarity between sentences to infer what instances constitute a code violation according to textual definitions. By exhibiting improved performances without a significantly different architect, it is clear that the descriptive road code definitions is an effective classification criterion, further answering *RQ1*.

SetFit and few-shot learning has not been an area of focus in classifying AEC texts as they struggle to meet the levels of reliability that AEC operations require. For example, as mentioned before, road maintenance texts in this case study are classified according to road codes set out in licensing terms. These are rigid classifications that are regulated, which demands a higher degree of accuracy. This case study demonstrates that the proposed similarity-based classification approach is able to outperform a state of the art few-shot text classifier without increasing the number of parameters used. By improving both overall and classwise performance, the proposed classifiers are more capable of producing high quality outputs that are better suited for the stringent nature of road inspection and maintenance, thus increasing the practicality of applying data-efficient deep-learning based text classifiers in engineering applications where there is limited data availability. Overall, the proposed classifier compares very favorably with SetFit, which helps in answering *RQ4*.

5.3.2. Improvements over traditionally fine-tuned models

Standard text classifiers are most commonly trained using traditional fine-tuning under the configuration shown in Fig. 2, and their architecture can vary by changing the pre-trained model. They are included in this case study as a baseline representing the most popular text classifier architecture with transformers. Three pre-trained models were chosen which all have roughly the same number of parameters as the proposed classifier to ensure comparability (Testing with much larger models is included in Section 6.2). They are represented on Fig. 12 in green.

Traditional fine-tuning based models are designed for settings where data availability is not an issue. Therefore, the performance of these models was first investigated by fine-tuning them over all available data with a training set of more than 300,000 inspection comments, allowing them to demonstrate their maximum performance. All three pre-trained models attained a high level of $accuracy$ of around 95% and around 0.9 in $f1$. However, when the proposed similarity-based classifier was fine-tuned using the same all-data training set, it was also able to achieve very similar results, even outperforming two of the three pre-trained traditional models. To further compare the maximum performances of the baseline models and the proposal, 14 shows the histograms of classwise $f1$ values of both the *bert-base-uncased* model and the best performing proposed model. There was an almost complete overlap, with most classes demonstrating very high (>0.85) $f1$ scores, reaffirming that the maximum performances of traditionally fine-tuned models are very similar to those of the proposed similarity-based model. This is a very interesting observation as the proposal of this classifier was motivated by its rapid domain adaptation in few-shot settings, not by its full-shot performance.

Traditional fine-tuning based methods differ from few-shot methods like SetFit in that they typically require a large volume of fine-tuning data to be performant. When data constraint is introduced in the few-shot settings, even though there are drops in performance across all models, those experienced by traditionally fine-tuned models were the biggest. At 10-shot, the proposed similarity-based classifier maintained around 2/3 of its maximum $f1$ value, while the traditionally fine-tuned models only retained between around 50% to 60% of their peaks. This discrepancy becomes greater as the number of examples per class is further reduced. At 1-shot, the proposed classifier still retained more than 50% of its maximum $f1$ compared to less than 10% for the traditionally fine-tuned models. As all models have roughly the same maximum performance determined during all-data training, this results in very significant differences in absolute performance in few-shot settings. At 1-shot, the proposal outperformed the traditional baselines by between 5 and 10 times. This lead was reduced to around 10% to 40% at 10-shot. It is apparent that the proposed classifier is more resilient to data scarcity than the traditional baselines, which is an expected result.

The above observations indicate that even though the proposed similarity-based classifier was designed for few-shot fine-tuning, it is also just as capable as other traditionally fine-tuned models when enough data is supplied. However, its additional few-shot characteristics allow it to retain much more of its maximum performance when faced with data-scarcity. This demonstrates the versatility of the proposed classifier as it can be effective at all levels of data availability. This resilience and versatility is particularly useful for applications like road maintenance, where there is significant data availability imbalance. For instance, there is a significant class imbalance in this case study dataset, where some road codes have tens of thousands of violation examples whereas others have around a hundred. The importance of a road code is not dependent on its frequency, thus requiring consistent performances across codes. By leveraging its resilience against data scarcity, it is able to retain a higher degree of performance for infrequent classes, contributing to this consistency. This indicates that the proposed similarity-based classifier is a better alternative than traditionally fine-tuned classifiers for classifying road maintenance texts, further answering *RQ4*.

5.4. Classwise performance analysis

To further explore factors that contribute to the observations made in the previous sections, the performances at the class and individual prediction levels are analyzed in the following sections. This provides the context for the prior observations, and aims at further answering *RQ1*.

5.4.1. Classification examples

The performance of the models on the class level can be inspected through analyzing the prediction results of the inspection comments. Examples of model predictions are listed in Table 11, where comments T are predicted to be violation instances of road codes P, with the actual violated codes being L. The top two rows show examples of a correct classification, while the last row shows an examples of an incorrect one. In some instances, keywords such as *"seal expansion joints"* are used in both the comment and code definition, making classification more straightforward. However, inspection comments might also paraphrase the code definitions, such as *"overdue for final restoration"* and *"restore ... within required time"*. The first two examples show that the similarity-based models can effectively infer these relationships.

However, in the third example, *"cover/street hardware"* was paraphrased as *"manhole"*, which likely confused the model into predicting an incorrect code which contained the exact word *"manhole"*. This indicates that the semantic meaning of the sentence was not fully captured by the similarity-based model. However, as both the predicted and correct codes concern issues with street hardware, the reasoning behind the model's classification decision is not unclear. As illustrated in Fig. 9 before, there is a high likelihood that even if an incorrect classification was made, the correct code would still be within the top prediction results and that the model effectively outputs a ranking of all codes according to similarity. This in itself can still be useful in a hybrid automation approach where the model selects the most likely road codes for a human operator to choose from upon data entry.

5.4.2. Performance imbalance between classes

By aggregating prediction results for each class, it is apparent classes have differing levels of performance. A consistent trend in the results is that $accuracy$ values are higher than $f1$ values. For example, the 10-shot Cross Entropy trained Bi-Encoder Only model achieved an $accuracy$ of 0.749, but an $f1$ score of only 0.608. This applies to all training scenarios and model choices. As micro-averaged $accuracy$ is the equivalent to micro-averaged $f1$, this indicates that the overall performance is skewed by high performances on frequently occurring classes.

Table 12 outlines the 3 best and worst performing classes and their number of occurrences in the dataset. As shown, the classes with the highest $f1$ scores all have thousands of examples, whereas the classes with the lowest have a bit more than 100. This confirms the discrepancy between the $accuracy$ and $f1$ results, in that having more examples coincides with having better performance, and that the performance of high performing classes are over-represented in $accuracy$ values.

5.4.3. Causes of performance imbalance

Though classes with fewer testing examples exhibit worse performance, it needs to be clarified that this is not a result from them having fewer training examples, as the number of examples per class is fixed during subsampling when creating the few-shot datasets. Given that all models have the same number of examples, no one class is over-represented when preparing models. Therefore, performance imbalance between classes indicate that the issue potentially lies within the dataset itself.

The dataset could be further processed and the validity of road codes could be re-examined. Fig. 15 shows the road codes that are predicted for all inspection comments with ground truth label *"Failure to obtain permit for temporary roadway pavement markings changes"*,

Table 11
Examples of classification results for inspection comments using the proposed similarity-based classifier. Two correct and one incorrect predictions are shown.

Inspection comment example T	Prediction of violated road code P	Correct violated road code L	Outcome
A.T.P.O. I observed the above respondent gas cut with temporary asphalt overdue for final restoration. permit expired 10.18.14. expired permit number above and gas box in cut on the sidewalk used for i d MUST APPEAR IN PERSON.	*Failure to permanently restore cut within required time*	*Failure to permanently restore cut within required time*	Correct
A/T/P/O I observed the respondent failed to seal expansion joints installed in steel faced curb. Respondent was notified by CAR#20217460371 on 11/17/21 to correct, but failed to do so. If not admitting the charge, you MUST APPEAR IN PERSON.	*Except as in NYC Administrative Code 19-152, failure to install or seal expansion joints as per subsection*	*Except as in NYC Administrative Code 19-152, failure to install or seal expansion joints as per subsection*	Correct
The respondent failed to repair "CON EDISON" marked manhole cover that is sunken approximately 2.5 inches with faded lettering. Corrective action request issued 7.13.20-CAR #20204560126 If not admitting the charge, you MUST APPEAR IN PERSON.	Failure to replace loose, slippery or broken utility maintenance hole (manhole) covers, castings, and other street hardware	Utility cover/street hardware not flush with surrounding area	Incorrect

Table 12
Three road codes with the highest and lowest classwise $f1$ score and their total number of examples in the dataset. Poorer results coincide with rarer classes.

Road code C	Total No.	Classwise $f1$
Highest		
Failure to use plating and/or decking that is skid-resistant in its entirety on roadway	4082	0.963
Construction material/equipment without proper reflective markings	3787	0.948
Failure to permanently restore cut within required time	62 884	0.927
Lowest		
Failure to restore in kind any parking or regulatory signs or supports	141	0.080
Failure to maintain a 5-foot pedestrian walkway on sidewalk	111	0.048
Failure to obtain permit for temporary roadway pavement markings changes	126	0.047

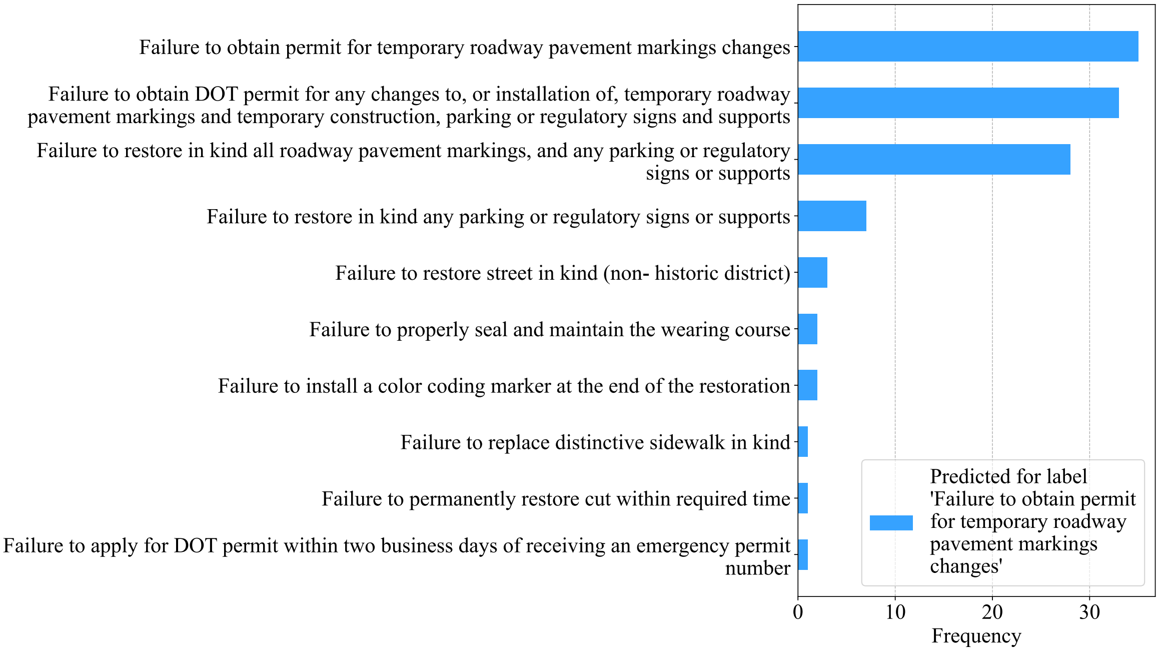


Fig. 15. Road codes predicted for examples with label *"Failure to obtain permit for temporary roadway pavement markings changes"*. The most probable incorrect predictions are all semantically similar to the ground truth label.

which is the class with the lowest $f1$ score. It shows that although the correct classification is the most common prediction, it is also frequently (roughly 2/3 of the time) misclassified into other classes. This can be attributed to code duplications and inherent inconsistencies in the dataset:

With respect to code duplications, the most likely predictions of misclassified examples are all semantically very similar to the actual label, where the majority references the approval or restoration of pavement markings. These are examples of codes which can be considered duplicates as a violation instance can easily fall within the bounds of all of them. Though there might still be minor technical differences, it is difficult to even manually infer the most fitting classification. As semantics are used as a direct classification criterion, the model's predictions are particularly sensitive to codes that are not well distinguished.

Regarding issues with inconsistencies, it is possible that many of these examples are incorrectly labeled or that the comments are not accurate. For example, the inspection comment *"I observed respondent failed to restore the bike lane paint after repairing a street light. Respondent was identified by color code marker in restoration. Respondent was notified prior by CAR#20217970039 and failed to correct the condition. If not admitting the charge, you MUST APPEAR IN PERSON"*. was labeled as an instance of *"Failure to obtain permit for temporary roadway pavement*

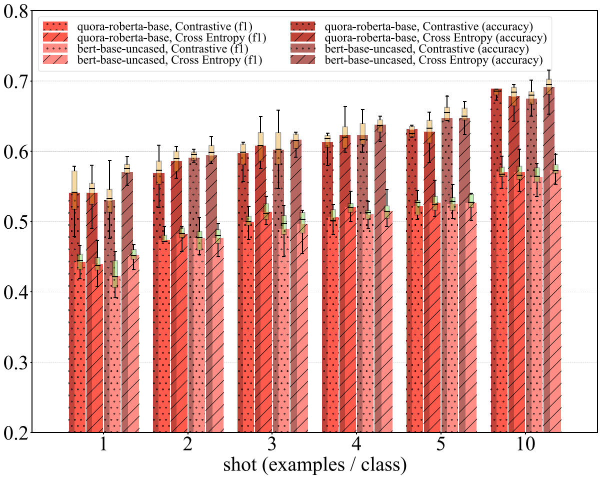


Fig. 16. Additional results in all few-shot scenarios for two cross encoder pre-trained models. Box plots are overlaid to summarize the results of the ten individual datasets per shot. Front bars represent $f1$ scores, rear bars represent $accuracy$. Dotted bars represent models trained using contrastive learning, while striped bars represent models trained using cross entropy. The results are very similar, indicating that the pre-trained model choice did not cause the discrepancy in performance between the Bi-Encoder Only and Reranked Cross Encoder architectures.

markings changes”, whereas *“Failure to restore in kind all roadway pavement markings, and any parking or regulatory signs or supports”* appears to be more appropriate.

These issues indicate that the difference in performance between classes can at least be partially attributed to the inherent quality of the dataset. This highlights the importance in ensuring that the quality of the dataset is high as it can affect how the model can perform. However, it can be seen that for well labeled and well-defined classes, the model exhibits a good level of performance.

6. Additional experimentation

In order to further consolidate the findings in Section 5, additional experimentation was performed which are discussed in the following sections. Section 6.1 further examines the cause of the inferior performances of the Cross Encoders compared to the Bi-Encoders. Section 6.2 compares the performance of the proposed classifier against state of the art baseline models with 70 times the number of parameters. Finally, 6.3 explains tests performed with ICL for this case study.

6.1. Experimentation with another Cross Encoder pre-trained model

As the results of the Reranked Cross Encoder classifiers are against the initial hypothesis, further experimentation was performed to determine the reasoning. One of the potential reasons discussed in Section 5.1.2 is the choice of pre-trained models which have different architectures and pre-training regimes that can impact their task adaptability.

In order to investigate the impact different pre-trained models have on the Cross Encoder configuration, additional tests on all few-shot datasets were performed (on top of the initial testing used to select the pre-trained models). In addition to *quora-roberta-base*, *bert-base-uncased* was also tested as a Cross Encoder reranker as it had the most comparable results in the initial tests. Results are plotted in Fig. 16 for the two pre-trained models. Consistent with initial testing, the difference in performance is minimal. Relatively large variations occur inconsistently and infrequently, such as in the 1-shot setting where the Contrastive Learning trained *bert-base-uncased* performed worse than the other models, though even that was only roughly a 5% difference.

It can therefore be shown that even the best performing Cross Ecnoder pre-trained models were not able to elevate the performance of the Reranked Cross Encoder classifiers, and that without a more suitable pre-trained model, Cross Encoders are not suitable for practical deployments of the proposed similarity-based classifier in AEC operations.

6.2. Testing with larger baseline models

All baseline and proposed models results in Section 5 are reported for models with roughly 100 million parameters, which ensures that results are comparable. These models are computationally inexpensive and can be run locally on a much wider range of computing hardware. For example, all models can be trained and used even on a laptop (tested on an M1 MacBook Pro with 32 GB of unified memory). However, to compare the performance of the proposed model with much more demanding state of the art baselines, an additional set of focused experimentation was performed. Fig. 17 plots the averaged metric values (along with box plots summarizing the performances of the individual subsampled datasets per shot) of three ∼7B models against the best proposed similarity-based classifier with 70 times fewer parameters. For completeness, the full numerical results are listed in Table 15 in Appendix.

Results indicate that the proposed architecture maintains the lead it has on all few-shot scenarios. Within the three larger baselines, *Llama3* performed the best, though its performance still lags behind the proposed similarity-based classifier despite being 70 times bigger. As with the other ∼100M traditionally fine-tuned baselines, the gap in performance is bigger in fewer shot settings. In a 1-shot case, the proposed classifier has 7 times the $f1$ score of *Llama3*. Even in a 10-shot setting, the proposal still leads by at least 34%.

This observation is likely due to the use of an uninitialized classification head required to produce class probabilities from the model outputs, as with other smaller traditionally fine-tuned models tested in Section 5. As they are the only untrained parameters, they require the most significant tuning, effectively acting as a bottleneck especially in few-shot settings.

As a result, it demonstrates that traditionally fine-tuned models are likely limited by its architecture and not the size of their pre-trained models. The results from Section 5.3.2 can therefore be generalized to all traditionally fine-tuned techniques, irrespective of the choice of pre-trained models. It also shows that through reformulation as a similarity-comparison task, computationally efficient models can be used to efficiently classify domain-specific road inspection texts in a complex technical setting better than a standard approach, making deployment much more scalable in terms of time costs and hardware requirements.

6.3. Testing with ICL

As ICL is another popular zero/few-shot methods that can leverage descriptive class definitions in the classification process through prompting, it has also been experimented with. *Llama-2-7b* was used for testing. ICL works by using a single prompt to supply descriptions of the task and all possible classes along with examples. However, with 83 classes, due to the context window constraint [37] and task ambiguity, it was difficult to generate consistent results:

Regarding the context window constraint, the maximum input token length is 4096. However, when providing just one example per class in a prompt, the token length is already more than 9000, meaning that examples could not fit. As a result, only zero-shot ICL with only the class definitions can theoretically be used. However, without providing examples, it is difficult to condition the model to output a classification. The total number of words included in all 83 class definitions is 921, which saturates the prompt. It is difficult for the model to attend fully

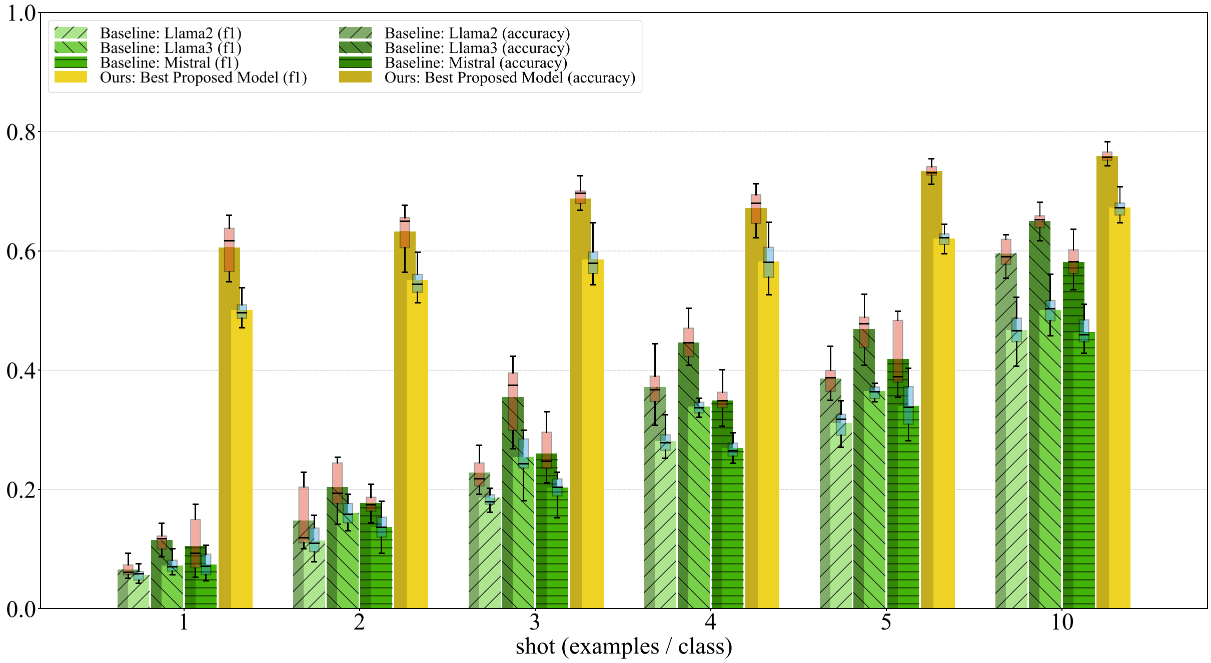


Fig. 17. Results for the additional experiments comparing the proposed ~100M similarity-based classifier with ~7B traditionally fine-tuned models. Box plots are overlaid to summarize the results of the three individual datasets per shot. Exact values and standard deviations are included in Table 15. Front bars represent $f1$, rear bars represent $accuracy$. Green bars represent ~7B baselines, yellow bars represent the best proposed classifier. The proposal outperforms all baseline models despite being 70 times smaller.

to the instructions and testing examples, as the former takes up most of the input prompt.

This is particularly an issue with smaller models, which is why base size models (~100 million) are rarely used for ICL. Even the 7 billion parameter model tested was not able to overcome this and output consistent results, and using even bigger models (Llama2 can have up to 70 billion parameters) would be very computationally expensive.

As a result, it was determined that ICL is not a suitable alternative in this case study. It both requires computationally expensive models and that it cannot effectively manage cases with many classes, which is a common scenario in many technical engineering text classification tasks like road maintenance.

7. Limitations and future work

Though this work has proposed an effective classifier that outperforms existing few-shot and traditionally fine-tuned models in a resource-constrained AEC settings, there are aspects which can be further explored to enhance the applicability of the proposal.

The followings are aspects which requires further investigations:

1. The proposed classifier relies heavily on the quality of the class representations, as they are used directly as the sole classification criterion. However, there are many possible class representations that can be adopted. In this case study, the representation was chosen to be descriptive class definitions that defines the instances which constitute a code violation. However, other forms of representation can be used which can potentially impact the model's performance. Comprehensive and consistent definitions can be beneficial, while ambiguous ones can limit it.
2. The dataset used in the case study contains imperfections. The distinctiveness of each road code and the quality of the annotation of the source data have affected the performance of the models for certain classes, which can impact overall metric values. This can introduce biases when interpreting absolute results, though in terms of relative performance the impact should be minimal.
3. The proposed classifiers utilize *SBERT*, which requires the choice of a pre-trained model. The choice of the models can impact the performance even with the same similarity-based classifier architecture and training regime as they have been prepared in different ways. As similarity-based classification is a unique reformulated approach to text classification, no pre-trained model was directly trained under the same objective, which requires the use of models trained for adjacent tasks. This can cause unexpected results, such as the Cross Encoders performing worse than the Bi-Encoders.
4. The case study focuses on the applicability of the proposed classifier in the context of road maintenance, which is an example of a technical domain akin to other AEC fields. However, as different domains and tasks generate texts in different formats, the findings of this work may not automatically apply to other AEC fields. For example, if a field relies on complex acronyms, the general knowledge that pre-trained models possess in natural language might not be fully applicable.
5. Though the few-shot results of the proposed models outperform all baselines in most cases, including baselines designed specifically for few-shot settings, the absolute performance of the model can still be improved. For example, at one example per class, the proposed model achieved an accuracy of almost 70%. This is high considering the level of resource constraint and the performances of alternative approaches, though it is still much less than the 95% achieved when all data is available for training. Part of the issue could be resolved by further clarifying class definitions and improving data annotation, though even then it would still likely contain errors, especially at extremely low-resource settings. This can limit the applicability of the proposed classifiers in very critical operations, and potentially require additional human supervision.

To overcome the above limitations, the following extensions are proposed:

1. Other forms of class representations can be explored to understand how the proposed classifiers perform in different circumstances. For example, simple key words could be used to

directly link critical concepts in classification in a focused manner. Long, descriptive paragraphs can also be used, utilizing detailed descriptions and examples to precisely define class contents. Structured language can also be used where each class representation is written in a predictable manner, allowing the models to reliably understand its content.

2. More datasets in different fields can be explored to assess how well the model can generalize. Even though it has been found to be an effective solution for road maintenance texts using road code definitions as a class representation, performance in other domains can differ due to the different ways texts are formatted and structured. For example, construction safety can be an interesting extension due to the existence of well-defined rules and regulations that can aid in providing comprehensive class representations. However, future experiments should focus on using high quality data that can ensure more representative resutls.
3. A new pre-trained model can be produced by training over publicly available AEC texts using similarity-based metrics as an training objective. This can both adapt the model to the domain-specific vocabulary of many AEC operations and also train it to perform well on similarity-adjacent tasks in the domain, such as similarity-based text classification.

8. Conclusion

Automating the classification of road maintenance texts can contribute to improving the efficiency of many maintenance operations and be beneficial to digitalization efforts such as the Road DT. However, as is the case with many other AEC operations, there is a considerable difficulty in collecting sufficient training data to employ state of the art deep learning techniques in a scalable manner. Improving the data-efficiency of these models can potentially be a means of overcoming this barrier to adoption by reducing the quantity of examples needed to create automation pipelines. Recognizing the inadequacies of current approaches, both in traditionally fine-tuned and few-shot techniques, the authors hypothesized that a similarity-based text classifier can be explored as an alternative approach to achieve this goal.

The proposed classifier functions by aligning textual inputs with pre-defined descriptive class definitions through encoding with *SBERT* and using the produced similarity scores directly as a classification criterion. The class with the highest semantic similarity with a textual input is selected as its classification prediction. This differs from existing approaches in that classes are represented informatively instead of through random indices, and that semantic similarity is directly used as a classification criterion instead of through logit outputs. Additionally, unlike previous work with zero-training approaches, *SBERT* is fine-tuned explicitly using classification as a training objective.

It has various advantages over approaches in other works. By not having a task-specific classification head, it is able to more rapidly adapt to different tasks than both traditionally fine-tuned models and few-shot techniques like SetFit. In addition, class representations used can act as a means of existing domain knowledge infusion that can pre-define the classes without the need for training data. Furthermore, compared to ICL, by not employing prompts, it both eliminates the need to define the task and creates the condition for accommodating tasks with many classes, allowing much smaller models to be used while also not being at risk of exceeding the maximum model input length.

Two similarity-based classifier architectures and two fine-tuning techniques are proposed. In terms of architecture, the Bi-Encoder Only and Reranked Cross Encoder approaches are proposed, where the it is hypothesized that the former would be quicker and the latter would be more accurate. They take advantage of the two main configurations of *SBERT*. In terms of the fine-tuning techniques, a classification-adapted contrastive learning approach and a similarity-adapted cross entropy approach are proposed. The former is more similar to standard *SBERT* fine-tuning for pairwise similarity tasks, whereas the latter is similar to standard single-text classification tasks.

Table 13
Definition of abbreviations used in this paper.

Abbrev.	Definition
ADMM	Asset Data Management Manual
DT	Digital Twin
NLP	Natural Language Processing
LLM	Large Language Model
GPT	Generative Pre-Trained Transformers
BERT	Bidirectional Encoder Representations from Transformers
AEC	Architecture, Engineering and Construction
SBERT	Sentence BERT
NYC	New York City
IDM	Information Delivery Manual
NER	Named Entity Recognition
TF-IDF	Term Frequency - Inverse Document Frequency
TLP	Technical Language Processing
FSL	Few-shot Learning
SetFit	Sentence Transformer Fine-Tuning
ICL	In-context Learning
PCA	Principal Component Analysis
t-SNE	t-distributed Stochastic Neighbor Embedding

A case study on applying the proposed classifier in a road maintenance text classification task was performed. The task utilized publicly available dataset from NYC Open Data that consists of more than 400,000 inspection records on roadworks. Each entry corresponds to an instance when one of 83 road code was violated when roadworks were undertaken. The simulated task involves using the free text inspection comments to infer the code that is being violated. This dataset is representative of datasets used in other road maintenance and wider AEC operations in that it makes use of technical language different from texts in a general context, and that it contains a large number of classes which is a common feature in many tightly regulated and complex fields.

The performance of the proposed classifiers was first compared. The theoretically less capable Bi-Encoder Only approach outperformed the Reranked Cross Encoder configuration in all settings, though the differences in performance are not very significant. This can be due to the differences in pre-trained models used and architecture-specific subsampling that was necessary to improve computational feasibility. It was also shown that by adopting the proposed fine-tuning strategies, models successfully adapted to the task context within a data-constraint setting. The Cross Entropy training approach produced better models than those trained with the adapted Contrastive Learning method, which is an expected result as cross entropy is widely used in standard classifier training. By further comparing zero-training results and visualizing the model's embedding space, it was shown that semantic similarity with descriptive road code definitions was effective in acting as a direct inspection comment classification criterion. Overall, these observations indicate that the proposed classifiers are all capable of being used as an effective text classifier in a complex road maintenance text classification setting, and that the Bi-Encoder Only configuration trained with Cross Entropy was the best performing model.

To understand the difference in performance resulting from the proposed classification algorithm, the proposed classifiers are evaluated against baseline models with roughly the same number of parameters. The proposal outperformed SetFit, a specialized few-shot learning text classifier, in all few-shot settings despite both being SBERT-based architectures, with the lead being bigger with less available data, up to 83% in a 1-shot setting. This indicates that the addition of descriptive road code definitions allowed the model to more rapidly adapt to the task. In terms of standard techniques that do not specialize in resource-constrained settings, he proposal also performed very similarly to traditionally fine-tuned models when all training data (>300,000) is available. More importantly, the proposal retained a much higher level of performance when faced with data constraints

Table 14
Averaged $f1$ and *accuracy* results along with standard deviations across ten independent subsampled datasets per shot for all ~100M baseline classifiers and proposed similarity-based classifiers. Standard deviations account for both the variances in training and few-shot dataset generation. All-shot refers to training with all available data. Con. and C. E. stand for Contrastive and Cross Entropy based models. This table is included for completeness.

Shots	Baselines				Similarity-based classifiers (**ours**)			
	Traditionally fine-tuned			Few shot	Bi-Encoder Only		Reranked Cross Encoder	
	BERT	RoBERTa	XLNet	SetFit	Con.	C. E.	Con.	C. E.
$f1$ scores								
1	$0.039_{0.013}$	$0.074_{0.019}$	$0.043_{0.012}$	$0.254_{0.018}$	$0.455_{0.020}$	**$0.464_{0.017}$**	$0.443_{0.015}$	$0.439_{0.019}$
2	$0.123_{0.019}$	$0.162_{0.018}$	$0.108_{0.016}$	$0.373_{0.011}$	$0.487_{0.018}$	**$0.497_{0.016}$**	$0.474_{0.020}$	$0.482_{0.018}$
3	$0.199_{0.031}$	$0.211_{0.024}$	$0.140_{0.030}$	$0.434_{0.015}$	$0.514_{0.017}$	**$0.528_{0.016}$**	$0.499_{0.014}$	$0.514_{0.014}$
4	$0.285_{0.038}$	$0.270_{0.011}$	$0.177_{0.063}$	$0.470_{0.014}$	$0.530_{0.012}$	**$0.540_{0.022}$**	$0.506_{0.013}$	$0.520_{0.012}$
5	$0.359_{0.035}$	$0.326_{0.016}$	$0.240_{0.036}$	$0.508_{0.023}$	$0.550_{0.018}$	**$0.560_{0.012}$**	$0.522_{0.014}$	$0.527_{0.015}$
10	$0.552_{0.019}$	$0.489_{0.019}$	$0.442_{0.024}$	$0.579_{0.016}$	$0.607_{0.019}$	**$0.608_{0.013}$**	$0.570_{0.015}$	$0.570_{0.017}$
All	**0.899**	0.890	0.893	N/A	0.838	0.898	0.776	0.844
Accuracy scores								
1	$0.058_{0.023}$	$0.092_{0.029}$	$0.066_{0.015}$	$0.304_{0.036}$	$0.560_{0.048}$	**$0.578_{0.038}$**	$0.541_{0.034}$	$0.541_{0.027}$
2	$0.150_{0.020}$	$0.195_{0.023}$	$0.124_{0.022}$	$0.443_{0.043}$	$0.600_{0.032}$	**$0.619_{0.038}$**	$0.569_{0.026}$	$0.586_{0.016}$
3	$0.249_{0.038}$	$0.260_{0.035}$	$0.163_{0.036}$	$0.525_{0.044}$	$0.630_{0.028}$	**$0.666_{0.022}$**	$0.597_{0.029}$	$0.608_{0.024}$
4	$0.345_{0.045}$	$0.331_{0.025}$	$0.217_{0.073}$	$0.576_{0.027}$	$0.668_{0.019}$	**$0.674_{0.029}$**	$0.613_{0.022}$	$0.623_{0.021}$
5	$0.445_{0.066}$	$0.410_{0.031}$	$0.292_{0.034}$	$0.619_{0.034}$	$0.673_{0.040}$	**$0.699_{0.018}$**	$0.631_{0.027}$	$0.628_{0.023}$
10	$0.673_{0.028}$	$0.600_{0.019}$	$0.547_{0.031}$	$0.691_{0.014}$	$0.733_{0.030}$	**$0.739_{0.019}$**	$0.689_{0.025}$	$0.678_{0.031}$
All	**0.954**	0.951	0.952	N/A	0.940	0.953	0.919	0.933

Table 15
Averaged *accuracy* and $f1$ results for ~7B traditionally fine-tuned models and the best proposed ~100M similarity-based classifier (Bi-Encoder Only classifier trained with Cross Entropy) along with standard deviations.

Shots	7B traditionally fine-tuned baselines						Best proposed (**ours**)	
	Llama2		Llama3		Mistral			
	acc.	$f1$	*acc.*	$f1$	*acc.*	$f1$	*acc.*	$f1$
1	$0.066_{0.014}$	$0.056_{0.010}$	$0.115_{0.020}$	$0.072_{0.012}$	$0.105_{0.042}$	$0.074_{0.019}$	$0.606_{0.038}$	$0.501_{0.023}$
2	$0.148_{0.052}$	$0.114_{0.013}$	$0.204_{0.037}$	$0.160_{0.020}$	$0.177_{0.019}$	$0.137_{0.022}$	$0.633_{0.034}$	$0.551_{0.028}$
3	$0.228_{0.033}$	$0.186_{0.026}$	$0.355_{0.056}$	$0.254_{0.036}$	$0.260_{0.039}$	$0.203_{0.028}$	$0.688_{0.024}$	$0.586_{0.033}$
4	$0.372_{0.039}$	$0.281_{0.032}$	$0.446_{0.030}$	$0.339_{0.015}$	$0.349_{0.026}$	$0.269_{0.019}$	$0.672_{0.030}$	$0.582_{0.036}$
5	$0.386_{0.027}$	$0.311_{0.024}$	$0.469_{0.033}$	$0.365_{0.015}$	$0.419_{0.055}$	$0.340_{0.039}$	$0.734_{0.011}$	$0.621_{0.017}$
10	$0.596_{0.025}$	$0.467_{0.031}$	$0.650_{0.018}$	$0.501_{0.029}$	$0.581_{0.026}$	$0.464_{0.025}$	$0.759_{0.012}$	$0.673_{0.020}$

compared to these traditional techniques, having a retention rate of 50% of $f1$ compared to less than 10% for the baselines in a 1-shot setting. These observations highlight the proposal's versatility in handling different levels of data availability and resilience against data scarcity. These are beneficial traits in that data availability across classes in AEC operations like road maintenance varies but consistency in performance is still often required.

To investigate the difference in performance against state of the art models without limiting the size of the baselines, a comparison was carried out with much larger ~7B pre-trained models. Results show that even with 70 times the number of parameters, the proposed model with 100 million parameters still demonstrated better few-shot performances. Using bigger models does not appear to improve the performance of the baselines, since the uninitialized classification head acts as a bottleneck in performance in very resource-constrained settings. Additionally, alternative few-shot approaches such as ICL was not able to be used in this case study due to the sheer number of classes and their complex descriptive definitions. These observations highlight the strength, efficiency, and scalability of the proposed approach in complex engineering applications, even compared to recent state of the art techniques.

However, it is important to mention that the there are limitations to this work. The use of descriptive class definitions and pre-trained models introduce variability of the models and can create issue with performance predictability. In addition, the case study dataset can be further analyzed to improve its quality and consistency to generate more representative results. Finally, despite the proposal improving over every baseline in few-shot settings, the absolute performance can still be improved to make practical deployments more feasible. The authors propose that these limitations be mitigated through a series of future works, including experimenting with alternative class representations to better understand how the models function, along with experimenting and training models with a larger variety of datasets.

To conclude, this work proposed a similarity-based text classifier that aimed at effectively utilizing descriptive definitions of road codes to classify road inspection comments using minimal training data. It outperformed competing approaches and demonstrated state of the art few-shot performances in the simulated resource-constrained road inspection text classification task. Therefore, by producing a scalable, efficient, and powerful text classifier, this work increased the feasibility of automating manual text analysis in complex engineering fields, and can no doubt be beneficial to many AEC applications where data availability is limited.

9. Table of abbreviations

Table 13 contains abbreviations used.

CRediT authorship contribution statement

Ching Yau Fergus Mok: Writing – original draft, Visualization, Validation, Software, Resources, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Lavindra de Silva:** Writing – review & editing, Supervision, Conceptualization. **Varun Kumar Reja:** Writing – review & editing, Supervision. **Stephen Green:** Writing – review & editing, Supervision. **Ioannis Brilakis:** Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge the EPSRC for funding this research through the EPSRC Centre for Doctoral Training in Future Infrastructure and Built Environment: Resilience in a Changing World (EPSRC grant reference number EP/S02302X/1) and the Digital Roads of the Future (EPSRC grant reference number EP/V056441/1).

Appendix. Full results tables

The following tables contain the full numerical results for experiments conducted in the case study.

Data availability

Data will be made available on request.

References

[1] Asset Data Management Manual, National Highways, 2021.
[2] Introduction to Digital Roads, National Highways, 2021.
[3] V.K. Reja, D. Davletshina, M. Yin, R. Wei, Q.F. Adam, I. Brilakis, F. Perrotta, A digital twin based approach to control overgrowth of roadside vegetation, in: V. Gonzalez-Moret, J. Zhang, B. García de Soto, I. Brilakis (Eds.), Proceedings of the 41st International Symposium on Automation and Robotics in Construction, International Association for Automation and Robotics in Construction (IAARC, Lille, France, 2024, pp. 661–668, http://dx.doi.org/10.22260/ISARC2024/0086.
[4] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al., Improving language understanding by generative pre-training, 2018, URL https://www.mikecaptain.com/resources/pdf/GPT-1.pdf.
[5] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2019, arXiv:1810.04805.
[6] M.P. Brundage, T. Sexton, M. Hodkiewicz, A. Dima, S. Lukens, Technical language processing: Unlocking maintenance knowledge, Manuf. Lett. 27 (2021) 42–46, http://dx.doi.org/10.1016/J.MFGLET.2020.11.001.
[7] Roads We Manage, National Highways, 2020, URL https://nationalhighways.co.uk/our-roads/roads-we-manage/. [Cited 16 October 2024].
[8] Gs 801 Asset Delivery Asset Inspection Requirements, Standards for Highways.
[9] Technical Note: Road Condition and Maintenance Data, Department for Transport.
[10] Y. Mo, D. Zhao, J. Du, M. Syal, A. Aziz, H. Li, Automated staff assignment for building maintenance using natural language processing, Autom. Constr. 113 (2020) 103150, http://dx.doi.org/10.1016/J.AUTCON.2020.103150.
[11] Traffic Signs Manual Chapter 1 Introduction, Department for Transport.
[12] Cm 125 Maintenance of Traffic Signs, Design Manual for Roads and Bridges, National Highways.
[13] K. Jeon, G. Lee, S. Kang, H. Roh, J. Jung, K. Lee, M. Baldwin, A relational framework for smart information delivery manual (idm) specifications, Adv. Eng. Inform. 49 (2021) 101319, http://dx.doi.org/10.1016/j.aei.2021.101319, URL https://www.sciencedirect.com/science/article/pii/S1474034621000720.
[14] R. Sacks, A. Kedar, A. Borrmann, L. Ma, I. Brilakis, P. Hüthwohl, S. Daum, U. Kattel, R. Yosef, T. Liebich, B.E. Barutcu, S. Muhic, Seebridge as next generation bridge inspection: Overview, information delivery manual and model view definition, Autom. Constr. 90 (2018) 134–145, http://dx.doi.org/10.1016/j.autcon.2018.02.033, URL https://www.sciencedirect.com/science/article/pii/S0926580517303977.
[15] R.S. Wilkho, S. Chang, N.G. Gharaibeh, Ff-bert: A bert-based ensemble for automated classification of web-based text on flash flood events, Adv. Eng. Inform. 59 (2024) 102293, http://dx.doi.org/10.1016/j.aei.2023.102293, URL https://www.sciencedirect.com/science/article/pii/S1474034623004214.
[16] S. Moon, G. Lee, S. Chi, Automated system for construction specification review using natural language processing, Adv. Eng. Inform. 51 (2022) 101495, http://dx.doi.org/10.1016/j.aei.2021.101495, URL https://www.sciencedirect.com/science/article/pii/S1474034621002445.
[17] J.P.U. Cadavid, B. Grabot, S. Lamouri, R. Pellerin, A. Fortin, Valuing free-form text data from maintenance logs through transfer learning with camembert, Enterp. Inf. Syst. 16 (6) (2022) 1790043, http://dx.doi.org/10.1080/17517575.2020.1790043, arXiv:https://doi.org/10.1080/17517575.2020.1790043.
[18] M.V. Ottermo, S. Håbrekke, S. Hauge, L. Bodsberg, Technical language processing for efficient classification of failure events for safety critical equipment, PHM Soc. Eur. Conf. 6 (2021) 9, http://dx.doi.org/10.36001/PHME.2021.V6I1.2792, URL http://www.papers.phmsociety.org/index.php/phme/article/view/2792.
[19] T. Sexton, M.P. Brundage, M. Hoffman, K.C. Morris, Hybrid datafication of maintenance logs from ai-assisted human tags, in: Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017 2018-January (2017), pp. 1769–1777, http://dx.doi.org/10.1109/BIGDATA.2017.8258120.
[20] Y. Wang, Y. Zhu, W. Xiong, C. Cai, A few-shot word-structure embedded model for bridge inspection reports learning, Adv. Eng. Inform. 62 (2024) 102664, http://dx.doi.org/10.1016/j.aei.2024.102664, URL https://www.sciencedirect.com/science/article/pii/S1474034624003124.
[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, 2023, arXiv:1706.03762.
[22] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2016, arXiv:1409.0473.
[23] L. Huang, W. Yu, W. Ma, W. Zhong, Z. Feng, H. Wang, Q. Chen, W. Peng, X. Feng, B. Qin, et al., A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions, ACM Trans. Inf. Syst. 43 (2) (2025) 1–55.
[24] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019, arXiv:1908.10084.
[25] Z. Xu, M. Iwaihara, Self-training involving semantic-space finetuning for semi-supervised multi-label document classification, Int. J. Digit. Libr. 25 (2023) 1–15, http://dx.doi.org/10.1007/s00799-023-00355-4, URL https://link.springer.com/article/10.1007/s00799-023-00355-4.
[26] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, D. Wierstra, Matching networks for one shot learning, 2017, arXiv:1606.04080.
[27] J. Snell, K. Swersky, R.S. Zemel, Prototypical networks for few-shot learning, 2017, arXiv:1703.05175.
[28] F. Sung, Y. Yang, L. Zhang, T. Xiang, P.H.S. Torr, T.M. Hospedales, Learning to compare: Relation network for few-shot learning, 2018, arXiv:1711.06025.
[29] T. Dopierre, C. Gravier, W. Logerais, A neural few-shot text classification reality check, 2021, arXiv:2101.12073.
[30] M. Garnelo, D. Rosenbaum, C.J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y.W. Teh, D.J. Rezende, S.M.A. Eslami, Conditional neural processes, 2018, arXiv:1807.01613.
[31] J. Requeima, J. Gordon, J. Bronskill, S. Nowozin, R.E. Turner, Fast and flexible multi-task classification using conditional neural adaptive processes, 2020, arXiv:1906.07697.
[32] J. Wang, K.-C. Wang, F. Rudzicz, M. Brudno, Grad2task: Improved few-shot text classification using gradients for task representation, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, J.W. Vaughan (Eds.), Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 6542–6554, URL https://proceedings.neurips.cc/paper_files/paper/2021/file/33a854e247155d590883b93bca53848a-Paper.pdf.
[33] L. Tunstall, N. Reimers, U.E.S. Jo, L. Bates, D. Korat, M. Wasserblat, O. Pereg, Efficient few-shot learning without prompts, 2022, arXiv:2209.11055.
[34] R.K. Helmeczi, Few-Shot Learning for Text Classification and Its Applications in Essay Scoring and Software Engineering (Master's thesis), Toronto Metropolitan University, 2023, URL https://rshare.library.torontomu.ca/articles/thesis/Few-Shot_Learning_for_Text_Classification_and_Its_Applications_in_Essay_Scoring_and_Software_Engineering/26866606?file=48866827.
[35] L. Loukas, I. Stogiannidis, P. Malakasiotis, S. Vassos, Breaking the bank with chatgpt: Few-shot text classification for finance, 2023, arXiv:2308.14634 URL https://arxiv.org/abs/2308.14634.
[36] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, T. Liu, B. Chang, X. Sun, L. Li, Z. Sui, A survey on in-context learning, 2024, arXiv:2301.00234 URL https://arxiv.org/abs/2301.00234.
[37] A. Milios, S. Reddy, D. Bahdanau, In-context learning for text classification with many labels, 2023, arXiv:2309.10954 URL https://arxiv.org/abs/2309.10954.
[38] A. Edwards, J. Camacho-Collados, Language models for text classification: Is in-context learning enough?, 2024, arXiv:2403.17661 URL https://arxiv.org/abs/2403.17661.
[39] P.V. Sappadla, J. Nam, E.L. Mencía, J. Fürnkranz, Using semantic similarity for multi-label zero-shot classification of text documents, in: The European Symposium on Artificial Neural Networks, 2016, URL https://api.semanticscholar.org/CorpusID:25074162.
[40] Z. Haj-Yahia, A. Sieg, L.A. Deleris, Towards unsupervised text classification leveraging experts and word embeddings, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 371–379, http://dx.doi.org/10.18653/v1/P19-1036, URL https://aclanthology.org/P19-1036.
[41] T. Schopf, D. Braun, F. Matthes, Lbl2vec: An embedding-based approach for unsupervised document retrieval on predefined topics, in: Proceedings of the 17th International Conference on Web Information Systems and Technologies, SCITEPRESS - Science and Technology Publications, 2021, http://dx.doi.org/10.5220/0010710300003058.

[42] T. Schopf, D. Braun, F. Matthes, Evaluating unsupervised text classification: Zero-shot and similarity-based approaches, in: Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval, NLPIR 2022, ACM, 2022, http://dx.doi.org/10.1145/3582768.3582795.

[43] D. Lavi, V. Medentsiy, D. Graus, Consultantbert: Fine-tuned siamese sentence-bert for matching jobs and job seekers, 2021, arXiv:2109.06501 URL https://arxiv.org/abs/2109.06501.

[44] H. Bekamiri, D.S. Hain, R. Jurowetzki, Patentsberta: A deep nlp based hybrid model for patent distance and classification using augmented sbert, Technol. Forecast. Soc. Change 206 (2024) 123536, http://dx.doi.org/10.1016/j.techfore.2024.123536, URL https://www.sciencedirect.com/science/article/pii/S0040162524003329.

[45] B.Y. Leão Imai, C. Mesquita Garcia, M.V. Rocha, A. Lameiras Koerich, A. De Souza Britto, J. Paul Barddal, Is it fine to tune? evaluating sentencebert fine-tuning for brazilian portuguese text stream classification, in: 2024 IEEE International Conference on Big Data (BigData), 2024, pp. 1765–1774, http://dx.doi.org/10.1109/BigData62323.2024.10825456.

[46] O. Khattab, M. Zaharia, Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020, arXiv:2004.12832.

[47] Nyc open data, 2023, NYC Open Data. URL https://opendata.cityofnewyork.us/. [Cited 01 July 2023].

[48] New York City Street Works Manual, New York City Department of Transport, 2025, URL https://streetworksmanual.nyc/chapter-four/street-construction-inspections-enforcement. [Cited 28 March 2025].

[49] Pretrained models - sentence-transformers documentation, 2024, URL https://www.sbert.net/docs/pretrained_models.html. [Cited 16 October 2024].

[50] S. Casola, I. Lauriola, A. Lavelli, Pre-trained transformers: an empirical comparison, Mach. Learn. Appl. 9 (2022) 100334, http://dx.doi.org/10.1016/j.mlwa.2022.100334, URL https://www.sciencedirect.com/science/article/pii/S2666827022000445.

[51] M. Henderson, P. Budzianowski, I. Casanueva, S. Coope, D. Gerz, G. Kumar, N. Mrkšić, G. Spithourakis, P.-H. Su, I. Vulić, T.-H. Wen, A repository of conversational datasets, 2019, arXiv:1904.06472 URL https://arxiv.org/abs/1904.06472.

[52] K. Lo, L.L. Wang, M. Neumann, R. Kinney, D. Weld, S2ORC: The semantic scholar open research corpus, in: D. Jurafsky, J. Chai, N. Schluter, J. Tetreault (Eds.), Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4969–4983, http://dx.doi.org/10.18653/v1/2020.acl-main.447, URL https://aclanthology.org/2020.acl-main.447.