Pretrained Domain-Specific Language Model for Natural Language Processing Tasks in the AEC Domain

Zhe Zheng[1], Xin-Zheng Lu[1], Ke-Yin Chen[1], Yu-Cheng Zhou[1], Jia-Rui Lin[1,*]

*Corresponding author, E-mail: lin611@tsinghua.edu.cn; jiarui_lin@foxmail.com

(1. Department of Civil Engineering, Tsinghua University, Beijing, 100084, China)

Abstract:

As an essential task for the architecture, engineering, and construction (AEC) industry, information processing and acquiring from unstructured textual data based on natural language processing (NLP) are gaining increasing attention. Although deep learning (DL) models for NLP tasks have been investigated for years, domain-specific pretrained DL models and their advantages are seldomly investigated in the AEC domain. Therefore, this work developed a large scale domain corpora and pretrained domain-specific language models for the AEC domain, and then systematically explores various transfer learning and fine-tuning techniques to explore the performance of pretrained DL models for various NLP tasks. First, both in-domain and close-domain Chinese corpora are developed. Then, two types of pretrained models, including static word embedding models and contextual word embedding models, are pretrained based on various domain corpora. Finally, several widely used DL models for NLP tasks are further trained and tested based on various pretrained models. The result shows that domain corpora can further improve the performance of static word embedding-based DL models and contextual word embedding-based DL models in text classification (TC) and named entity recognition (NER) tasks. Meanwhile, contextual word embedding-based DL models significantly outperform the static word embedding-based DL methods in TC and NER tasks, with maximum improvements of 8.1% and 3.8% in the F1 score, respectively. This research contributes to the body of knowledge in two ways: 1) demonstrating the advantages of domain corpora and pretrained DL models and 2) opening the first domain-specific dataset and pretrained language models named ARCBERT for the AEC domain. Thus, this work sheds light on the adoption and application of pretrained models in the AEC domain.

Keywords:

Construction; natural language processing (NLP); transfer learning; pretrained BERT model; text classification (TC); named entity recognition (NER); language representation

1 Introduction

The architecture, engineering, and construction (AEC) industry is a significant driver of economic activity around the world. Currently, the AEC industry is undergoing a significant transformation from conventional labor-intensive methods to automation through the adoption of information and automation technologies (Wang et al., 2020; Wu et al., 2022a; Liao et al., 2021). To enhance the automation and informatization of the AEC industry, it is essential to make full use of all kinds of data, including structured data and unstructured data. However, over 80% of data in the AEC sector are unstructured, and most of them are texts, which are difficult to handle (Wu et al., 2022a). Hence, many efforts have been devoted to natural language processing (NLP) technologies, which aim to extract and understand text data automatically and intelligently. Recently, NLP technologies have been increasingly adopted in the AEC sector and have been applied in many applications, including filtering information, organizing documents, expert systems, and automated rule checking (ARC) (Hassan et al., 2021; Fuchs, 2021; Wu et al., 2022a). Among them, the ARC is one of the main application scenarios (Wu et al., 2022a), which involves almost all typical NLP tasks in the AEC domain. ARC aims to automate the compliance check process of a building design with applicable regulatory texts. The ARC process can be subdivided into four stages: 1) rule interpretation, 2) building model preparation, 3) rule execution, and 4) reporting checking results (Eastman et al., 2009; Ismail et al., 2017). Among the four stages, rule interpretation, which translates regulatory text into a computer-processable format, is the most vital and complex stage needing further investigation (Ismail et al., 2017).

In recent years, with emerging deep learning (DL) techniques and open datasets for model training, it has become possible for deep NLP-based methods to achieve a more comprehensive understanding of regulatory texts (Fuchs, 2021). As a result, many NLP methods have been further developed and improved based on DL techniques (Fuchs, 2021). Generally, the DL-based NLP methods used in AEC mainly include the following two tasks:

(1) Text classification (TC) divides texts into different groups. Typical applications of TC include a) recognizing relevant sentences in a regulatory text corpus, thereby avoiding inefficiency and errors in downstream tasks (e.g., NER) resulting from unnecessary processing of irrelevant text (Zhou & El-Gohary, 2016; Song et al., 2018), and b) identifying construction site accidents from large quantities of documents (Cheng et al., 2020).

(2) Named entity recognition (NER) detects semantic elements in a sentence and is also known as information extraction (IE) in the AEC domain. Typical applications of NER include a) identifying and extracting the words and phrases in the relevant sentences to interpret rules into computer-processable representations (Zhou et al., 2020; Zhang & El-Gohary, 2021; Moon et al., 2021) and b) automatic construction and extensions of knowledge graphs (Leng et al., 2019; Wu et al., 2022b).

Compared to traditional machine learning models, DL methods have significantly more parameters and typically need a larger scale of data for training (Zhang & El-Gohary, 2021). However, there is still a lack of unified semantic labels in the AEC domain, and few public training datasets are available for further investigation. As a result, DL-based approaches require highly expensive manual effort to prepare sufficient training datasets (Xu & Cai, 2021). Due to the lack of labeled training datasets, an unsupervised learning technique called model pretraining is introduced, which can learn rich syntactic and semantic patterns from large-scale textual datasets collected from the internet. Thus, it is possible to improve the performance of DL methods by transfer learning techniques that lack large-scale labeled datasets. For example, the bidirectional encoder representation from transformers (BERT) proposed by Devlin et al. (2018), a widely used NLP model based on DL, is pretrained on a large-scale corpus with 3,300 M words (Devlin et al., 2018). In this way, the proposed BERT model can learn hidden knowledge from large-scale datasets by pretraining and further help various downstream tasks by transfer learning (Fang et al. 2020). Applications in bioinformatics (Mohan et al., 2021) and ARC (Fang et al., 2020; Zhou et al., 2020) have illustrated the advantages of pretrained language representation models such as BERT.

However, as pointed out by Sun et al. (2019), the widely used BERT models are pretrained on the general-domain corpus, which has a different data distribution from the domain corpora targeted to a certain domain. That is, directly adopting BERT models pretrained on a general corpus may underestimate the contribution of domain-specific knowledge hidden in the domain corpora, which is quite valuable for training DL models for NLP tasks in the AEC domain (Zhang & El-Gohary, 2021). Unfortunately, to the best of our knowledge, there are no publicly available domain-specific corpora related to the AEC domain. No domain-specific transformer-based model is available for downstream NLP tasks in the AEC domain. Consequently, it is still unclear how domain-specific corpora impact the performance of DL-based methods for NLP tasks in the AEC domain.

To address this need, this work develops an open-source domain corpus and systematically investigates the performance of the various domain corpora and transfer learning strategies on DL models. Because the ARC processes involve almost all typical NLP tasks in the AEC domain, this work takes the ARC tasks as typical cases. First, two transfer learning strategies are discussed in detail. Second, two domain corpora datasets were constructed. Third, various domain corpora and transfer learning strategies are investigated for various DL models and the downstream NLP tasks (i.e., TC and NER). Finally, the ARCBERT models are obtained utilizing the developed domain corpora based on the BERT model (Devlin et al., 2018), which achieved state-of-art results on the typical downstream NLP tasks of ARC.

The remainder of this paper is organized as follows. Section 2 reviews the related work and highlights the potential research gaps. Section 3 describes the transfer learning strategies for AEC. Section 4 illustrates the dataset development. Sections 5 and 6 illustrate and analyze the results of the experiment. Section 7 discusses the advantages and contributions of this research and notes the limitations. Finally, Section 8 concludes this research.

2 Overview of related studies

2.1 DL-based natural language processing methods in the AEC domain

DL models are composed of multiple processing layers that enable the learning of data representations with multiple levels of abstraction and have shed light on sequential data such as text (LeCun et al., 2015). The main drawback of traditional machine learning models is the reliance on manual feature engineering, which is very time-consuming (Wu et al., 2022a). DL models can extract features automatically from training text data. With the development of DL techniques and open datasets for model training, it is possible for deep NLP-based methods to achieve a more comprehensive understanding of regulatory texts (Fuchs, 2021). DL approaches have outperformed traditional approaches in four major tasks in NLP, including classification, matching, translation, and sequence labeling (Li, 2017). Thus, with the advent of DL methods, NLP methods for AEC have been further developed. Generally, the DL-based NLP methods used in AEC mainly include two tasks: (1) TC and (2) NER. Therefore, the existing research on the TC and NER tasks are reviewed in the following two subsections (Sections 2.1.1 & 2.1.2). Moreover, DL models can also be divided into two main types: (1) contextual word embedding-based DL models (e.g., BERT-based models) and (2) static word embedding-based DL models (e.g., CNN-based models using static word embedding). For the static word embedding, each word has a single vector, regardless of context. The static word embedding model can be obtained utilizing the skip-gram models (Mikolov et al., 2013a). For contextual word embedding, the word vectors are sensitive to the context in which they appear (Ethayarajh, 2019). The transformer-based model (e.g., BERT) can create contextualized word embedding utilizing the masked language model. The structure and pretrained methods of contextual word embedding models differ from those of static word embedding models. Thus, the existing studies are categorized and summarized in Table 1, according to the tasks and models.

2.1.1 DL-based text classification

The text classification (TC) task aims to recognize relevant sentences from large quantities of documents and assign them to one or more predefined categories (Manning & Schutze, 1999) to facilitate downstream tasks. DL-based methods formulate the TC problem as a typical classification task, where each sentence/document is assigned a label. To deal with unstructured text data, the widely used DL-based methods first represent the sentences as vectors using word embedding techniques and then classify the vectors using DL models. Many DL models for TC, including TextCNN (Chen, 2015), TextRNN (Liu et al., 2016), TextRCNN (lai et al., 2015), DPCNN (Johnson et al., 2017), transformers (Vaswani et al., 2017), and BERT (Devlin et al., 2018), have been proposed in recent years. In the AEC domain, there has been a growing body of research efforts using DL-based methods to solve TC problems. For example, Tian et al. (Tian et al., 2021) classified on-site construction reports into six categories based on TextCNN. The result shows that the performance of TextCNN outperformed the traditional machine learning models. Cheng et al. (Cheng et al., 2020) deployed the gated recurrent unit (GRU) model for construction site accident classification. Zhong et al. (Zhong et al., 2020) developed a CNN model to classify accident narratives. Li et al. (Li et al., 2020) deployed a fastText-based classification model for analyzing construction accident claims. Fang et al. (Fang et al., 2020) classified near-miss information contained within safety reports using the BERT model, which achieved the best performance. To date, most of the studies have utilized static word embedding-based DL models for TC.

2.1.2 DL-based named entity recognition

The named entity recognition (NER) task aims to identify and extract the structured information in the relevant sentences for downstream usages, including interpreting rules into computer-processable formats (Zhou et al., 2020; Zhang & El-Gohary, 2021) and construing knowledge graphs (Leng et al., 2019). DL-based methods formulate the NER problem as a sequence labeling task, where each word or phrase is assigned a label (Zhang & El-Gohary, 2021). Similar to the methods for TC, the unstructured text data are also represented into vectors before model training and prediction. The typical DL models for NER include LSTM (Greff et al., 2017), BiLSTM-CRF (Huang et al., 2015), LSTM-CNN-CRF (Ma & Hovy, 2016), and BERT. In the AEC domain, there are now an increasing number of studies that focused on DL-based methods for the NER task. For example, the LSTM-based models are designed to identify semantic information elements for compliance checking purposes (Zhong et al., 2020; Moon et al., 2021; Zhang & El-Gohary, 2021; Feng & Chen, 2021; Moon et al., 2022) and for semiautomated knowledge graph construction (Leng et al., 2019). Zhou et al. (Zhou et al., 2020) utilized the pretrained BERT model for extracting predefined semantic labels in Chinese regulatory texts. Subsequentially, they formalized the extracted elements into language-independent pseudocodes. To date, most of the studies have utilized static word embedding-based DL models for NER.

Table 1 DL-based natural language processing methods

Tasks/DL model types	Static word embedding-based DL models	Contextual word embedding-based DL models

TC	Cheng et al., 2020;
	Zhong et al., 2020;
	Li et al., 2020;
	Tian et al., 2021
		Fang et al., 2020

NER	Leng et al., 2019;
	Zhong et al., 2020;
	Zhang & El-Gohary, 2021;
	Moon et al., 2021;
	Feng & Chen, 2021;
	Moon et al., 2022;
		Zhou et al., 2020

2.2 Transfer learning methods and domain corpora for AEC

DL-based approaches require highly expensive manual effort to prepare enough training datasets (Xu & Cai, 2021). However, in the AEC domain, there are not sufficient training datasets or publicly available domain corpora for many domain-specific NLP applications, such as TC and NER. To address this problem, various transfer learning strategies that can leverage labeled data from other domains have been proposed. Transfer learning is used to improve a model from one domain by transferring information from a related domain (Weiss et al., 2016). Deep transfer learning methods can be classified into four categories: instance-based methods, mapping-based methods, network-based methods, and adversarial-based methods (Tan et al., 2018). Instance-based methods utilize instances in the source domain by a specific weight adjustment strategy (e.g., TaskTrAdaBoost technology (Yao & Doretto, 2010)). Mapping-based methods map instances from the source domain and target domain into a new data space, where instances from two domains are similar (e.g., transfer component analysis (Pan et al., 2010)). Network-based methods, which are the most popular transfer techniques in NLP, refer to the reuse of the partial network pretrained in the source domain. Network-based methods can be mainly divided into two types, i.e., contextual word embedding models and static word embedding models. Adversarial-based methods use adversarial technology to find transferable features that are both suitable for two domains (Tan et al., 2018).

2.3 Research gaps

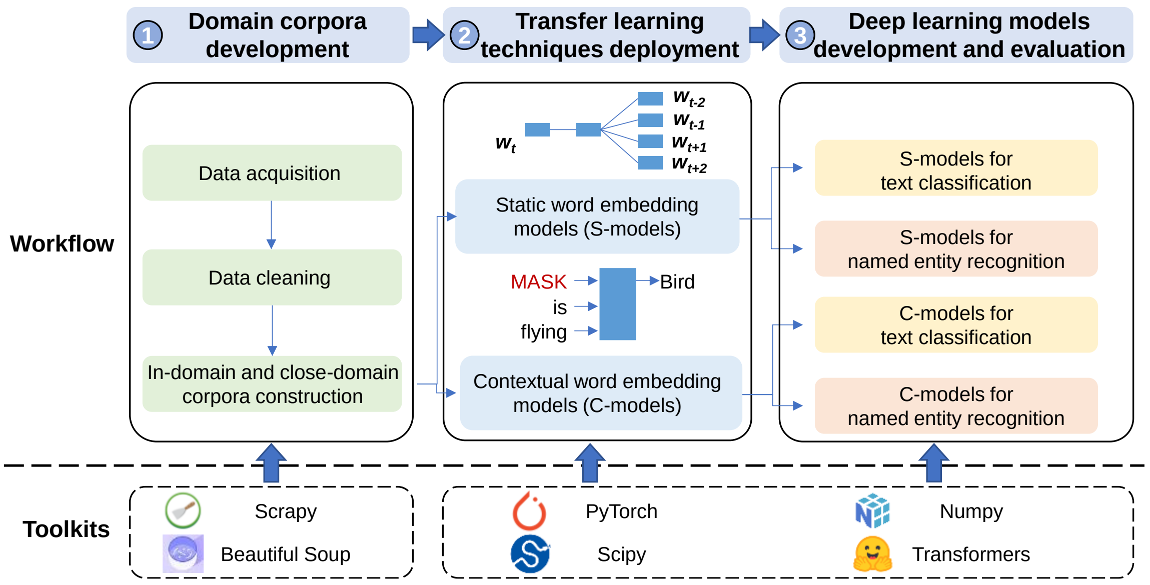
In the AEC domain, some research efforts have explored the use of DL-based models in AEC downstream NLP tasks (e.g., TC and NER). Despite the importance of these efforts, there are two knowledge gaps that this paper aims to address. First, the existing studies focused on only training or fine-tuning DL models on their own application-specific datasets. Few studies have explored transfer learning techniques to overcome the challenge of lacking labeled training data. Second, there are few open publicly available domain corpora prepared for transfer learning techniques in the AEC domain, which hampers the use of transfer learning methods. In conclusion, there is no systemic discussion of how to improve the performance of DL-based models utilizing domain corpora and transfer learning techniques, which require less human effort than labeling training datasets.

To address these gaps, this work aims to handle the following four scientific issues: (1) constructing AEC domain corpora to facilitate transfer learning usages, (2) exploring whether domain corpora are more efficient than general corpora for enhancing model performance, (3) investigating the value of domain corpora with various data distributions, and (4) investigating the performances of various transfer learning strategies for widely used DL models in the AEC domain.

3 Methodology

To improve the performance of DL models on NLP tasks such as TC and NER without increasing the effort of manual annotation, we systematically illustrate and analyze two typical domain corpus-based transfer learning models (i.e., static word embedding models and contextual word embedding models) for DL models, in Sections 3.1 and 3.2, respectively.

The investigation workflow of various domain corpus-enhanced transfer learning methods is shown in Fig. 1. The proposed workflow consists of three parts: (1) domain corpora development, where two domain corpora are constructed (Section 4.1), (2) transfer learning techniques deployment, where two transfer learning techniques are deployed based on the constructed domain corpora (Section 4.2), and (3) DL models development and evaluation, where several DL models are developed and evaluated for TC and NER tasks (Section 5). The core toolkits used in the three parts are also listed at the bottom in Fig. 1. The workflow is conducted based on the Python script.



1 Domain corpora development
2 Transfer learning techniques deployment
3 Deep learning models development and evaluation
Workflow
Data acquisition
Data cleaning
In-domain and close-domain corpora construction
w_t
w_{t-2}
w_{t-1}
w_{t+1}
w_{t+2}
Static word embedding models (S-models)
MASK
is
flying
Bird
Contextual word embedding models (C-models)
S-models for text classification
S-models for named entity recognition
C-models for text classification
C-models for named entity recognition
Toolkits
Scrapy
Beautiful Soup
PyTorch
Scipy
Numpy
Transformers

Fig. 1 The investigation workflow of various domain corpus-enhanced transfer learning methods

3.1 Domain corpus enhanced for static word embedding-based DL methods

3.1.1 Basic training workflow

The basic training and prediction flow of static word embedding-based DL models is shown by the blue arrow in Fig. 2. Words themselves cannot be directly understood by machines and inputted into deep neural networks. So, word embedding should be conducted first to represent the sequence of words into a vector matrix representation that can be recognized and computed by deep neural networks. Therefore, the basic training flow consists of two main steps: (1) a word embedding model, which represents each word in a text into a low-dimensional real number vector, thus obtaining the text's vector representation, and (2) a DL model, which further encodes the text's vector representation and then outputs the target prediction to complete the downstream tasks (i.e., TC or NER). As illustrated in Section 2.1.1 and 2.1.2, different downstream tasks will use different DL models and architectures. The goal of this section is to explore the effect of transfer learning strategies with pervasive applicability on different DL models. Therefore, the introduction of different architectures of DL models is not described in detail. The DL model architectures used in the experiments in Section 5.3 can be found in the related references.

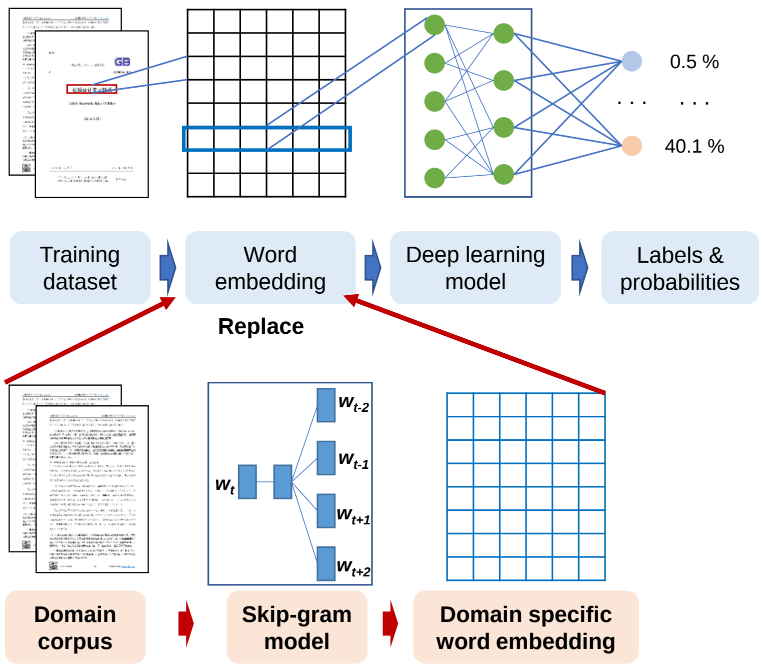


Fig. 2 Domain corpus-enhanced static word embedding-based DL methods

3.1.2 Domain corpus enhanced workflow

For the existing studies in the AEC domain, the word embedding models are either random initial (Tian et al., 2021) or pretrained on general corpora (Zhang & El-Gohary, 2021). The general corpora usually consist of the internet news corpora and the encyclopedia corpora (e.g., Wikipedia corpus (Wikipedia, 2021a) and Sogou news (Sogou, 2021)), where AEC-specific domain-related texts are rare. This may cause the model to have difficulty understanding the relevance of the AEC domain vocabulary, thus reducing the DL models' performance on downstream tasks. Some research efforts have shown that domain-specific word embedding models can capture domain-specific terms better than general terms (Wang et al., 2018). A natural idea is to pretrain the word embedding model first with the target domain data.

The flow of domain corpus-based transfer learning methods for static word embedding-based DL models is shown by the red arrow in Fig. 2. The flow consists of two main steps: (1) AEC domain-specific word embedding models, which are pretrained on domain corpora that contain more domain-specific terms. The word embedding model and the corresponding pretraining method are described in Section 3.1.3. And (2) word embedding model replacement, where the word embedding models pretrained on the general corpora are ousted by the AEC domain-specific word embedding model. The AEC domain-specific model can be used by loading the parameters of the pretrained embedding model.

3.1.3 Word embedding model and pretraining method

Because the performance of the continuous bag-of-word (CBOW) model and the skip-gram model is close (Mikolov et al., 2013a; Zhao et al., 2017), we choose the skip-gram model for the word embedding model. The skip-gram model inputs a single center word and then predicts the contextual words. As shown in Fig. 3, when the center word is "thickness", the model is expected to predict the corresponding surrounding words "The”, "firewall’s", "should", and "be". The input of the skip-gram model only needs raw text because the sliding window will be applied, as shown in Fig. 3.

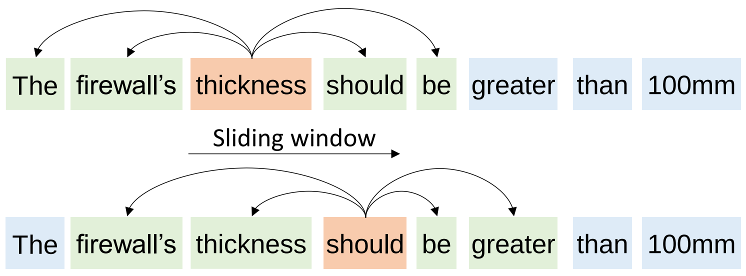


Fig. 3 Skip-gram model with window size 2

The skip-gram model can be mathematically formulated as follows. Let $\{w_1,\ w_2,\ w_3, \dots ,\ w_N\}$ be the center words, the objective is to maximize the conditional probabilities of contextual words given the center words, i.e.,

$$max \sum_{n=1}^{N} \left[\sum_{c \in C(w_n)} \log\left(logP(c|w_n;\theta)\right) \right] \tag{1}$$

where $C(w_n) = \{w_i,\ n - window \leq i \leq n + window\ and\ i \neq n\}$, and the $window$ means the window size, which indicates the number of contextual words that should be considered near the center word. The θ is the parameters of the word embedding matrix to be optimized. $P(c|w)$ is the conditional probability, and is often defined by softmax function, i.e.,

$$P(c|w) = \frac{e^{\theta_c^T \theta_w'}}{\sum_{c \in V} e^{\theta_c^T \theta_w'}} \tag{2}$$

where θ' and θ are the input and output word embeddings, respectively. Further details of the skip-gram model can be found in Mikolov et al. (Mikolov et al., 2013a) and Zhao et al. (Zhao et al., 2017).

In this work, the pretrained domain-specific word embedding model takes the domain corpus as the training dataset, and the domain corpus is established in Section 4.1. To accelerate the training process, negative sampling techniques are used (Mikolov et al., 2013b). The detailed hyper-parameter settings are illustrated in Section 4.2.1. Note that the word embedding pretraining task is a typically unsupervised task and does not require additional manual labeling.

3.2 Domain corpus enhanced for contextual word embedding-based DL models

3.2.1 Basic fine-tuning workflow

The basic training and prediction flow of the contextual word embedding-based DL model (i.e., transformer-based models) is shown by the blue arrow in Fig. 4. In Section 3.2, we take the BERT model (a kind of transformer-based model) as an example to illustrate. The flow consists of two main steps: (1) word embedding and encoding, which first embeds all tokens of the input sentence and then uses a pretrained BERT model to encode the input embeddings to contextual representations, and (2) prediction via classifier layers, where the contextual representations are then input to the last few classifier layers to obtain the prediction result.

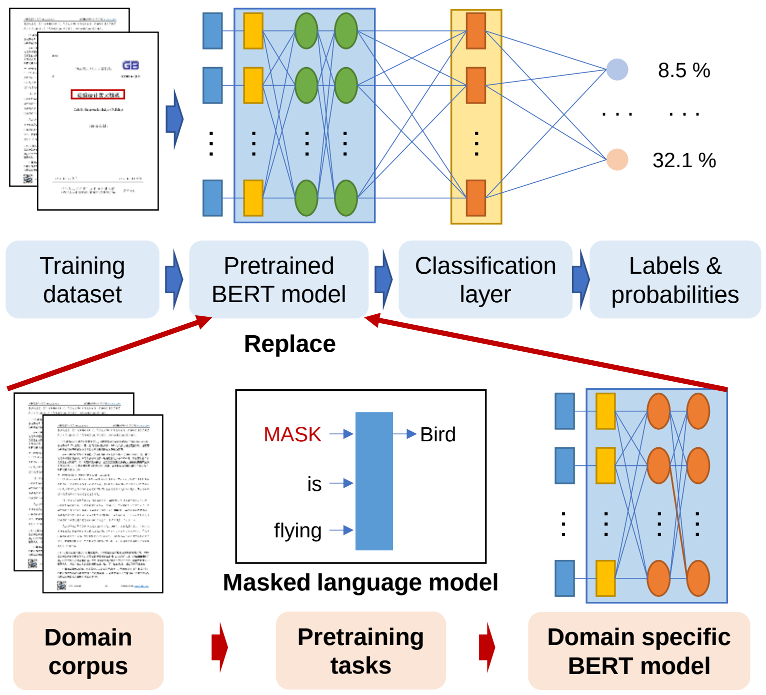


Fig. 4 Domain corpus-enhanced contextual word embedding-based DL methods

The Self-attention mechanism in the transformer allows BERT to model many downstream tasks by just swapping out the inputs and outputs. For the fine-tuning process for each task, we can simply plug in the task-specific inputs and outputs into BERT and adjust all the parameters end-to-end (Devlin et al., 2018). As shown in Fig. 5(a), for the TC task, the input is a single sentence, and the output is the class label of the sentence. As shown in Fig. 5(b), for the NER task, the input is a single sentence, and the output is the label of the semantic element represented in a special tagging format (e.g., BIO (beginning, inside, outside) tagging format).

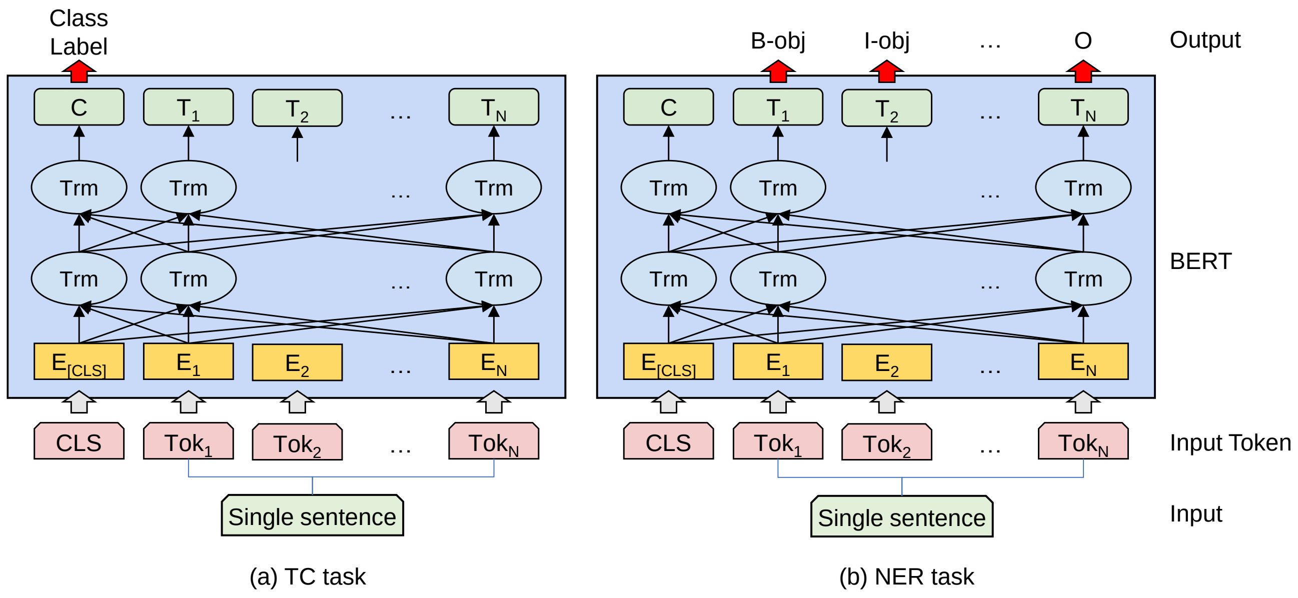


Fig. 5 Illustrations of fine-tuning process of BERT on TC and NER Tasks.

3.2.2 Domain corpus enhanced workflow

However, the widely used transformer-based models are pretrained in the general-domain corpus (Sun et al., 2019), which has a different data distribution from that of the target domain. When the training datasets are very limited, the transformer-based models pretrained in the general-domain corpus still struggle to obtain satisfactory results. A natural idea is to further pretrain transformer-based models with the target domain data.

The flow of domain corpus-based transfer learning methods for transformer-based models is shown by the red arrows in Fig. 4. The flow consists of two main steps: (1) AEC domain-specific BERT-based model pretraining based on a masked language model (Devlin et al., 2018), and (2) pretrained BERT model replacement. After the pretraining task, the AEC domain-specific BERT models can be used to replace the BERT model pretrained in the general domain. After the replacement, the fine-tuning and prediction tasks are the same as before mentioned. The BERT model and masked language model are described in the following section.

3.2.3 BERT model and masked language model

The BERT model mainly consists of (1) word embedding layer, (2) encoding layer, and (3) classification layer, as shown in Fig. 6. The embedding layer sums the token embedding, segment embedding, and position embedding, which represents the input words into vector representation for BERT input representation. The encoding layer consists of twelve identical bidirectional transformers, which computes a better contextual representation based on the self-attention mechanism. The classification layer takes the contextual representation from the encoding layer as input and predicts the target results, which often consist of a fully connected layer and a softmax layer. Further details of the structure of the BERT model can be found in Devlin et al. (Devlin et al. 2018).

To further pretrain the BERT model on the AEC domain corpus, the masked language model is used, as shown in Fig. 6. The masked language model randomly masks a few words in a sentence, and then inputs the masked sentence into BERT to predict the true words, which is similar to the cloze test. The [CLS] is a special symbol added in the front of every input example. Then, the CrossEntropy loss is used to calculate the loss, i.e.,

$$Loss = -\sum_{c=1}^{C} y_c \log{(p_c)} \tag{3}$$

where, y_c means the true one-hot encoded labels. And p_c means the output probability distributions for each output ‘token’. The loss function means the difference between the probability distributions for each output ‘token’ and the true label. We use the Transformer python library (Hugging Face, 2019) to implement the masked language model, which can automatically process the plain text into the desired masked form. The hyperparameters of the masked language model are illustrated in Section 4.2.2. Note that, the masked language model is a typically unsupervised task, which does not require additional manual labeling efforts.

After pretraining the BERT model, the parameters of the word embedding layer and encoding layer (i.e., the yellow blocks in Fig. 6) have been optimized. Then for different tasks, users can simply replace the input and the output (Devlin et al. 2018), and then fine tune the model, as illustrated in Section 3.2.1.

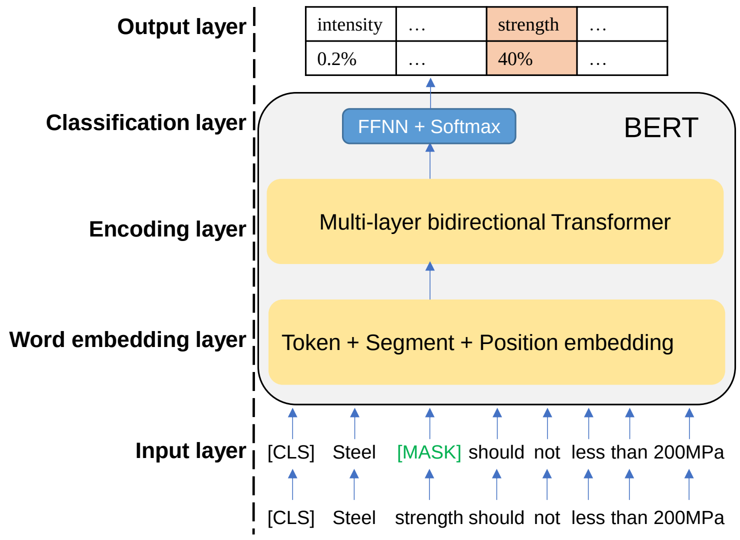

Output layer
Classification layer
Encoding layer
Word embedding layer
Input layer
intensity
…
strength
…
0.2%
…
40%
…
FFNN + Softmax
BERT
Multi-layer bidirectional Transformer
Token + Segment + Position embedding
[CLS] Steel [MASK] should not less than 200MPa
[CLS] Steel strength should not less than 200MPa

Fig. 6 Architecture of BERT model for masked language model

4 Domain corpora development and pretraining configuration

4.1 Domain corpora development

To analyze the performance of transfer learning techniques utilizing various domain corpora, this work collects a large number of domain corpora and subsequently constructs in-domain corpora and close-domain corpora, respectively. The civil engineering regulatory texts are the object of analysis for the ARC downstream tasks. Therefore, the in-domain corpora mainly contain regulatory texts. The close-domain corpora contain the AEC domain text, such as the definition of various terms in the AEC domain. The construction processes of the in-domain corpus and close-domain corpus are described in Sections 4.1.1 and 4.1.2, respectively.

The statistical information of the constructed corpora is summarized in Table 2. Note that although the number of lines in the close-domain corpus is smaller than that of the in-domain corpus, the number of texts in the close-domain corpus is close to that of the in-domain corpus. Because in the close-domain corpus, one line contains more characters. The developed domain corpora can be found in https://github.com/SkydustZ/AEC-domain-corpora/tree/main/domain%20corpus.

4.1.1 In-domain corpora construction

The data distribution of the in-domain data is the same as the training data distribution. In this research, the TC and NER training and validation datasets (Section 5.1) are from Chinese regulatory texts; thus, the in-domain corpora consist of the Chinese regulatory texts crawled from the *website of the Chinese codes* (Soujianzhu, 2021). The construction process consists of (1) data acquisition and (2) data cleaning. First, a total of 396 design codes are crawled based on scrapy (Myers & McGuffee, 2015). Only the text of the crawled design codes is utilized to form the in-domain corpus. The obsolete design codes are also taken into account to build a sufficiently large in-domain corpus. Data cleaning was then performed. The data cleaning consists of two steps. Step 1 content filtering filters the sentences that do not have Chinese chars, and the sentences not related to regulatory. In Chinese codes, if one sentence doesn't start with a numerical serial number, it's an irrelevant sentence. Step 2 sentence splitting splits the long regulatory texts according to sentence-ending punctuation (. , !, or ?). Finally, the cleaned in-domain corpus contained 126,433 lines and 10,895,634 Chinese characters.

4.1.2 Close-domain corpora construction

The data distribution of close-domain data is close to the training data distribution. In this research, the close-domain corpora consist of texts from the pages of the civil engineering category in Wikipedia (Wikipedia, 2021b) and texts from the civil engineering section in *Encyclopedia of China* (Encyclopedia of China Publishing House, 2009).

For the Wikipedia corpora, the construction process consists of (1) data acquisition and (2) data cleaning. First, data acquisition is performed based on scrapy (Myers & McGuffee, 2015). The pages and subpages of the civil engineering category in Wikipedia contain various civil engineering-related terms and their corresponding definitions. However, the subpages of the civil engineering category contain lots of irrelevant pages, such as terms that introduce the games and traveling. So, the filtering word list is established first. Once the terms contain words in the list, the crawling script will pass the page and its subpages, which greatly narrows down the crawling scope, reduces the crawling time and irrelevant terms. Then, the seed page is set to "https://zh.wikipedia.org/wiki/Category:土木工程(i.e., civil engineering in Chinese)" and the crawling script starts. Finally, a total of 11,189 terms are crawled from the pages.

Afterward, data cleaning was performed to sift the texts that were not relevant to the AEC domain via manually identifying the irrelevant terms. After data cleaning, a total of 10,488 terms and their corresponding descriptions were collected. The definition of one concept is stored in one line; thus, 10,488 lines are collected.

The *Encyclopedia of China* corpus is obtained by parsing the text and performing data cleaning. Then, a total of 16,238 lines of text are obtained. The above two are aggregated together to form the close-domain corpus, which contains 26,727 lines and 12,899,562 Chinese characters in total.

Table 2 Statistical information of the constructed domain corpora

	Number of lines	Number of Chinese characters
In-domain corpus	126,433	10,895,634
Close-domain corpus	26,727	12,899,562

4.2 Pretraining configuration

4.2.1 Pretraining static word embedding models

For the static word embedding model training, the Wikipedia Chinese corpora (Wikipedia, 2021a) are used together with the domain corpora, as the domain corpora are too small to contain all common words. Therefore, four word embedding models are trained: (1) the general model, which is trained only using the Wikipedia Chinese corpora (Wiki corpora for short) and is set as the control group; (2) the in-domain model, which is trained using the Wiki and the in-domain corpora; (3) the close-domain model, which is trained using the Wiki and the close-domain corpora; and (4) the mix-domain model, which is trained using the Wiki, the in-domain, and the close-domain corpora.

To train the word embedding models, Chinese word segmentation is performed on the above corpora using the jieba toolkit. The jieba toolkit is an open-source Chinese segmentation application in Python language. Subsequently, the segmented corpora are used to train the word embedding model utilizing the skip-gram method (Mikolov et al., 2013a) with the negative-sampling technique (Mikolov et al., 2013b). As mentioned in Section 3.1.3, the skip-gram method uses the headword to predict the context words around it with a similar basic theory. The negative-sampling technique is a more efficient way of deriving word embeddings. The word embedding models are trained with a dimension of 300, which is widely used by many researchers (Li et al., 2018; Wang et al., 2018) and with a negative-sampling number of 5. According to Zhao et al. (Zhao et al., 2017), for training word embeddings, we set the local window size to 2. And high-frequency words are removed with subsampling at the degree of 1×10^{-5}.

4.2.2 Further pretraining the contextual word embedding models

For the further pretraining of the contextual word embedding models, three domain corpora are considered: (1) the in-domain corpora, (2) the close-domain corpora, and (3) the mix-domain corpora, which combines the in-domain and close-domain corpora. Besides, to explore the influence of the amount of in-domain data, another two in-domain corpora are established by randomly splitting the in-domain corpora, including (4) 1/3 in-domain corpora, which contains one-third of the in-domain corpora, and (5) 1/5 in-domain corpora, which contains one-fifth of the in-domain corpora. The control group is the model without further pretraining, i.e., pretrained on only the general corpora.

In addition, we investigated the further pretraining performance of various contextual word embedding models. Two contextual word embedding models are chosen: (1) bert-base-chinese (Hugging Face, 2019) and (2) ERNIE (Sun et al., 2019). The bert-base-chinese model, which is pretrained on the Wikipedia Chinese corpora, is the most common BERT-based model for tasks in Chinese. The ERNIE model was published by Baidu and is pretrained based on the Baidu Chinese corpora and entity-level masking and phrase-level masking techniques, and it performs better than the BERT model on some downstream tasks (Sun et al., 2019).

For the further pretraining task, the masked language model is used, as mentioned in Section 3.2.3. We use a 15% probability of masking each token during model pre-training. For the models utilizing in-domain, close-domain, and mix-domain corpora, they are further pretrained with a learning rate of 5×10^{-5}, a batch size of 4, and an epoch of 3. For the models utilizing 1/3 in-domain corpora and 1/5 in-domain corpora, the hyperparameters are the same as the former ones except for the number of epochs. To ensure that the total amount of corpus during pretraining is consistent, the epochs of 9 and 15 are used for the models utilizing 1/3 in-domain corpora and 1/5 in-domain corpora, respectively.

Finally, five further pretrained BERT models and five further pretrained ERNIE models are obtained, as listed in Table 3.

Table 3 Further pretrained models

Base model/ Corpus	In-domain	Close-domain	Mix-domain	1/3-in-domain	1/5-in-domain

BERT	ARCBERT-Large	CivilBERT	MixBERT	ARCBERT-Mid	ARCBERT-Small
ERNIE	ARCERNIE-Large	CivilERNIE	MixERNIE	ARCERNIE-Mid	ARCERNIE-Small

5 Experiments and analysis

5.1 Datasets

We evaluate our approach and domain corpora on two datasets, including the regulatory TC dataset and NER dataset. The two datasets are briefly introduced as follows.

5.1.1 Regulatory text classification dataset

The regulatory TC dataset developed by the authors (Zheng et al., 2022) is used in this work. Seven text categories, including direct, indirect, method, reference, general, term, and others, were defined by the authors based on the computability of a text. The dataset distribution is shown in Fig. 7.

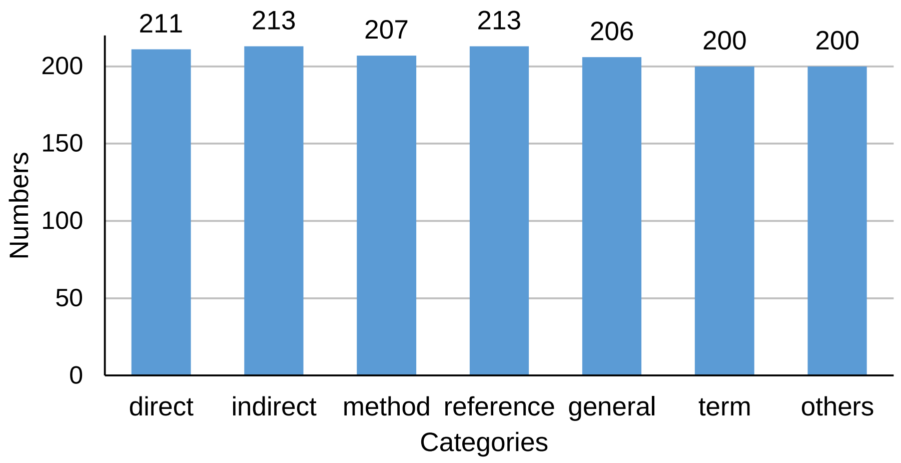


Fig. 7 Distribution of the text classification dataset

5.1.2 Named entity recognition dataset

The NER dataset developed by Zhou et al. (Zhou et al., 2020) is used in this work. Seven semantic labels, including obj, sobj, prop, cmp, Rprop, ARprop, and Robj, were defined by the authors (Zhou et al., 2020) based on the hierarchical structure of the objects and attributes in the BIM model. Then, the gold standard for semantic labeling was developed (Zhou et al., 2020). The dataset includes 611 sentences and has a total number of 4336 semantic elements. A repository containing the dataset was established on GitHub at https://github.com/Zhou-Yucheng/auto-rule-transform. The dataset distribution is shown in Fig. 8.

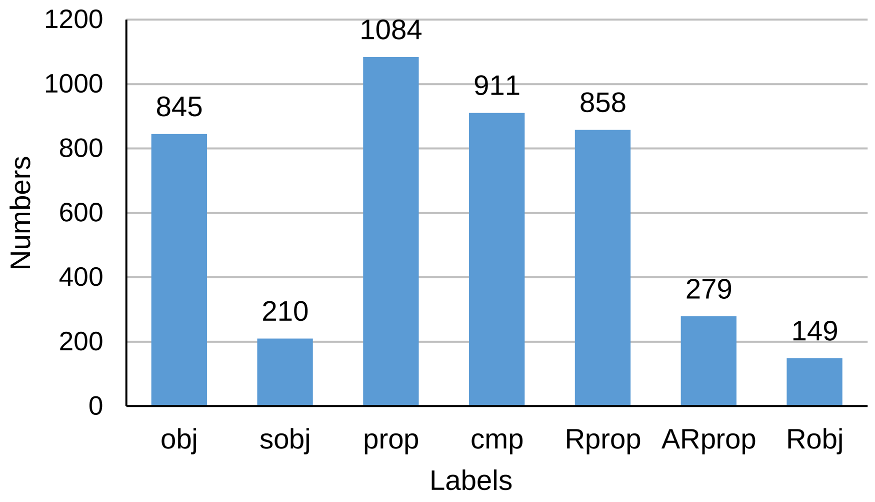


Fig. 8 Distribution of the named entity recognition dataset

5.2 Evaluation performance metrics

To measure the result, the model predictions are compared with the gold standard. For the TC models, the widely used weighted average F1 score (weighted F1) is selected. For the NER models, considering the datasets are highly unbalanced, the macro average F1 score (macro F1) is selected.

First, the precision (P), recall (R), and F1 score (F1) are calculated for each semantic label:

$$P = N_{correct}/N_{labeled} \tag{4}$$

$$R = N_{correct}/N_{true} \tag{5}$$

$$F_1 = 2PR/(P+R) \tag{6}$$

where $N_{\{correct,labeled,true\}}$ denotes the number of {model correctly labeled, model labeled, true} elements for each label.

Then, the weighted average F1 score is calculated to represent the overall performance for TC models (n_i denotes the number of elements of the i-th semantic label):

$$\text{Weighted } F_1 = \left(\sum_i n_i F_{1,i}\right) / \sum_i n_i \tag{7}$$

And the macro average F1 score is calculated to represent the overall performance for NER models (m denotes the number of types of semantic labels):

$$\text{Macro } F_1 = \left(\sum_i^m F_{1,i}\right)/m \tag{8}$$

5.3 Experiments

To investigate the various domain corpora and transfer learning techniques for the NLP tasks (i.e., TC & NER) in the ARC domain, several experiments are conducted in this section. The two datasets are randomly split into training, validation and test datasets at a 0.8: 0.1: 0.1 ratio, where the training dataset is used to train and update the DNN model, and the validation dataset is used to test the performance of the model and chose the best combination of the hyperparameters and the best model, and the test dataset is used for the final evaluation.

5.3.1 Experiment 1: domain corpus-enhanced static word embedding for TC

This experiment aims to investigate the performance of domain-specific pretrained word embeddings on TC tasks. The four pretrained word embedding models in Section 4.2.1 are combined with various DL-based TC models. Six types of DL-based TC models are employed to perform the experiment: (1) TextCNN (Chen, 2015), (2) TextRNN (Liu et al., 2016), (3) TextRNN with attention (TextRNN-Att), (4) TextRCNN (lai et al., 2015), (5) DPCNN (Johnson et al., 2017), and (6) Transformers (Vaswani et al., 2017). The above models are trained with a max epoch of 100, which is far more than the epoch that the optimal model requires. In addition, the effects of various learning rates and batch sizes are considered via grid search on learning rates of 2×10^{-3}, 1×10^{-3}, 5×10^{-4}, 2.5×10^{-4}, and 1×10^{-4} and on batch sizes of 32, 64, and 128. The optimizer is Adam, other parameters are defaulted except for the initial learning rate. The results are shown in Table 4.

Table 4 shows that in the TC task, there are slight improvements in most of the deep models using domain-specific word embedding except for the TextCNN and the DPCNN model. The close-domain word embedding model is suitable for RNN-based models (i.e., TextRNN and TextRNN-Att) and the transformer model. The mix-domain word embedding model is suitable for the TextRCNN model and the transformer model. However, there is no word embedding model that works for all models. This result is in line with the findings of Wang et al. (Wang et al., 2018) in the biomedical informatics domain. There is no consistent global ranking of word embeddings for TC applications in the ARC domain. However, considering the improved performance of most deep models, the use of domain-specific word embeddings could be considered when employing deep models for TC in the ARC domain. Especially when employing the RNN-based model, word embedding models trained on the close-domain corpus could be considered.

Table 4 Weighted F1 score on the text classification datasets

Model/Word embedding	General	In-domain	Close-domain	Mix-domain
TextCNN	**86.26%**	**86.26%**	84.60%	84.60%
TextRNN	66.00%	77.04%	**77.41%**	**77.41%**
TextRNN-Att	77.21%	75.64%	**81.76%**	73.55%
TextRCNN	79.13%	79.13%	81.64%	**84.93%**
DPCNN	**83.42%**	**83.42%**	74.50%	74.50%
Transformers	71.19%	71.19%	**71.74%**	**71.74%**

5.3.2 Experiment 2: domain corpus-enhanced static word embedding for NER

This experiment aims to investigate the performance of domain-specific pretrained word embeddings on NER tasks. The four pretrained word embedding models in Section 4.2.1 are combined with various DL-based NER models. Four types of DL-based NER models are employed to perform the experiment: (1) LSTM (Greff et al., 2017), (2) BiLSTM, (3) BiLSTM-CRF (Huang et al., 2015), and (4) BiLSTM-CNN-CRF (Ma & Hovy, 2016). The models are trained with a max epoch of 1000, which is far more than the epoch that the optimal model requires. In addition, the effects of various learning rates and batch sizes are considered via a grid search on learning rates of 1.5×10^{-2}, 1×10^{-2}, 5×10^{-3}, 1×10^{-3}, and 5×10^{-4} and on batch sizes of 8, 16, and 32. The optimizer is Adam with a weight decay rate equaling 1.0×10^{-8}. Besides, other parameters are defaulted except for the initial learning rate. The results are shown in Table 5.

Table 5 shows that in the ARC domain NER task, there are improvements on all deployed deep models when using domain-specific word embedding models instead of using the general one. All the models using the mix-domain word embedding performed better than the models using the general word embedding. The best score is achieved by using the close-domain word embedding model. Similar to the situation of the TC task, there is no consistent global ranking of word embeddings for NER applications in the ARC domain. However, considering the improvements on all models, the use of domain-specific word embeddings, especially the mix-domain one, could be considered when employing deep models for NER in the ARC domain.

Table 5 Macro F1 score on the named-entity recognition datasets

Model/Word embedding	General	In-domain	Close-domain	Mix-domain
LSTM	76.26%	73.64%	76.30%	**76.54%**
BiLSTM	76.26%	74.75%	76.15%	**76.54%**
BiLSTM-CRF	77.23%	74.93%	**78.03%**	77.66%
BiLSTM-CNN-CRF	68.77%	**77.46%**	74.79%	76.19%

5.3.3 Experiment 3: further pretrained contextual word embedding-based models for TC

This experiment aims to investigate the performance of further pretrained contextual word embedding-based models for TC tasks. Ten further pretrained models in Section 4.2.2 are applied to the TC task. The models are fine-tuned with a max epoch of 100, which is far more than the epoch that the optimal model requires. In addition, the effects of various learning rates and batch sizes are considered via grid search on learning rates of 1×10^{-5}, 3×10^{-5}, 5×10^{-5}, and 7×10^{-5}, and on batch sizes of 32, 64, and 128. The optimizer is BertAdam with a weight decay rate equaling 0.01. Besides, other parameters are defaulted except for the initial learning rate. The results are shown in Table 6.

Table 6 shows that in the ARC domain TC task, all further pretrained models on in-domain (i.e., ARCBERT-Large and ARCERNIE-Large), mix-domain (i.e., MixBERT and MixERNIE), and 1/3-in-domain corpora (i.e., ARCBERT-Mid and ARCERNIE-Mid) perform better than the original ones. Among them, the ARCBERT-Large model achieved the global best weighted F1 score of 94.42%. However, the performances of the further pretrained models on the close-domain corpora (i.e., CivilBERT and CivilERNIE) are worse than the original ones. Although the ARCERNIE-Small model achieved the best score among the ERNIE models, the ARCBERT-Small model performed worst among the BERT models. For the BERT model, the larger the amount of in-domain corpora, the better the model performs. Generally, further pretraining on in-domain corpora can bring better performance than further pretraining on close-domain corpora. This experiment illustrates that further pretraining on in-domain corpora is very useful to improve the performance of BERT for the TC task.

Table 6 Weighted F1 score on the text classification datasets

BERT	ARCBERT-Large	CivilBERT	MixBERT	ARCBERT-Mid	ARCBERT-Small
88.04%	**94.42%**	87.24%	89.80%	90.20%	87.16%
ERNIE	ARCERNIE-Large	CivilERNIE	MixERNIE	ARCERNIE-Mid	ARCERNIE-Small
87.25%	91.20%	86.41%	90.45%	88.86%	**91.40%**

5.3.4 Experiment 4: further pretrained contextual word embedding-based models for NER

This experiment aims to investigate the performance of further pretrained contextual word embedding-based models for NER tasks. Ten further pretrained models in Section 4.2.2 are applied to the ARC NER task. The models are fine-tuned with a max epoch of 50, which is far more than the epoch that the optimal model requires. In addition, the effects of various learning rates and batch sizes are considered via grid search on learning rates of 1×10^{-5}, 3×10^{-5}, 5×10^{-5}, and 7×10^{-5}, and on batch sizes of 8, 16, and 32. The optimizer is AdamW with a weight decay rate equaling 0.01. Besides, other parameters are defaulted except for the initial learning rate. The results are shown in Table 7.

Table 7 shows that in the ARC domain NER task, the further pretrained BERT models on all kinds of domain corpora performs better than the original one. Moreover, the ARCBERT-Small model achieves the global best macro F1 score of 81.8%. However, the performances of the further pretrained ERNIE models are equal to or worse than the original one. This experiment illustrates that the further pretrained BERT model is useful for improving the performance of the ARC NER task.

Table 7 Macro F1 score on the named-entity recognition datasets

BERT	ARCBERT-Large	CivilBERT	MixBERT	ARCBERT-Mid	ARCBERT-Small
76.4%	79.6%	79.6%	79.5%	79.8%	**81.8%**
ERNIE	ARCERNIE-Large	CivilERNIE	MixERNIE	ARCERNIE-Mid	ARCERNIE-Small
80.4%	78.7%	78.7%	**80.4%**	79.8%	80.1%

5.4 Summary

5.4.1 Text classification task

For the TC task in the ARC domain, the performances of various models on the test dataset are shown in Fig. 9. Almost all models except the TextCNN model and the DPCNN model are enhanced by the domain corpus and transfer learning techniques. Among all models, the TextRNN model achieves the most notable performance improvement, which increases the weighted F1 score by 11.4%. The second is the further pretrained BERT model, which increases the weighted F1 score by 6.4%. Meanwhile, the BERT model further pretrained on in-domain corpora (i.e., ARCBERT-Large) achieves the highest performance among all models, with a weighted F1 score of 94.4%. Therefore, for the TC task in the ARC domain, the in-domain corpus can be used to train domain-specific word embedding models or further pretrain the BERT and ERNIE models to improve the performance.

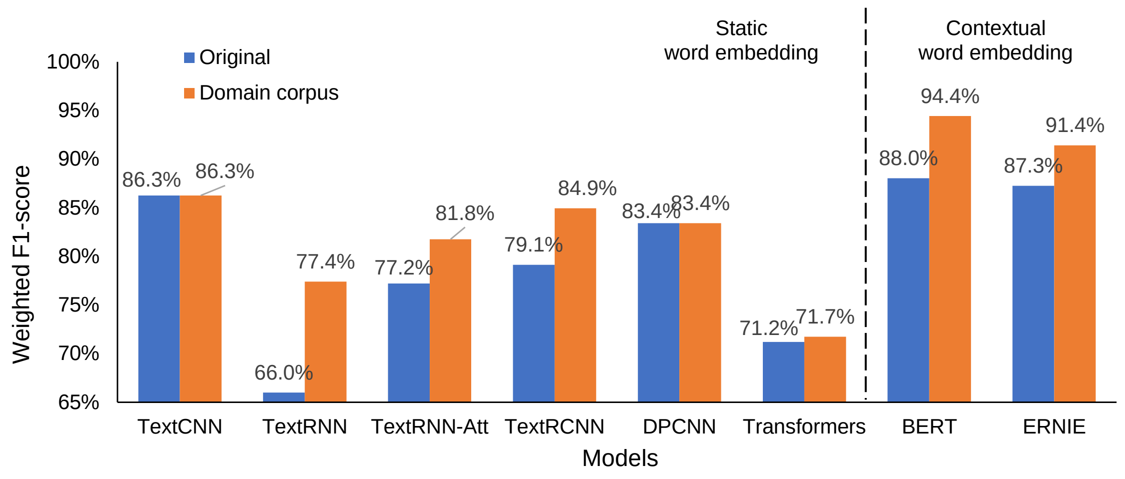


Fig. 9 Performance of various models on the TC task

5.4.2 Named entity recognition task

For the NER task in the ARC domain, the performances of the various models on the test dataset are shown in Fig. 10. Almost all models except for the ERNIE model are enhanced by the domain corpus and transfer learning techniques. Among all models, the BiLSTM-CNN-CRF model achieves the most notable performance improvement, which increases the macro F1 score by 8.7%. The second is the further pretrained BERT model, which increases the macro F1 score by 5.4%. Meanwhile, the BERT model further pretrained on 1/5-in-domain corpora (i.e., ARCBERT-Small) achieves the highest performance among all models, with a macro F1 score of 81.8%. Therefore, for the NER task in the ARC domain, the domain corpus can be used to train domain-specific word embedding models or further pretrain the BERT model to improve the performance.

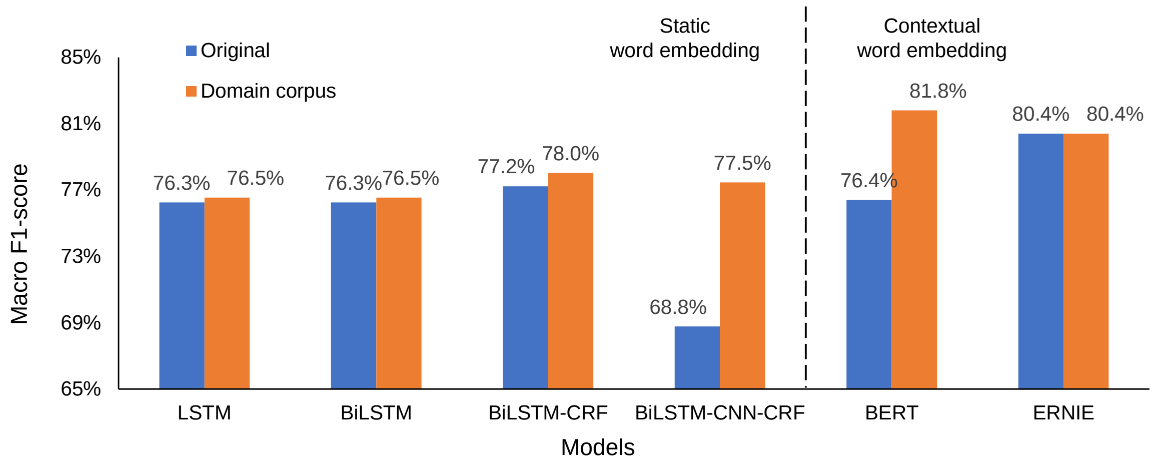


Fig. 10 Performance of various models on the NER task

5.4.3 Theoretical explanation

In general, transformer-based (i.e., BERT and ERNIE) models perform better than static word embedding-based DL models. A possible reason is that transformer-based models could consider context-related information hidden in the text. While other models use static word embedding, where each word has a single vector, regardless of context. All senses of a polysemous word share the same representation. While utilizing the masked language model, the transformer-based model can create contextualized word representations, where the word vectors are sensitive to the context in which they appear (Ethayarajh, 2019). Therefore, using contextualized word representations better allows the machine to obtain a full understanding of the text.

6 Discussion

This work offers a leading initiative by developing the first domain corpora and enhancing DL-based transfer learning techniques for various NLP tasks in the AEC domain. To improve the performance of DL models on ARC downstream tasks without increasing the effort of manual annotation, we systematically illustrate and analyze two typical domain corpus-based transfer learning methods for DL models. In comparison to the previous effort, this work contributes to the body of knowledge on four main levels:

(1) Word embedding models and further pretraining techniques are typical unsupervised tasks that can enhance the models with unlabeled domain corpora and eliminate extensive labeling efforts. To systematically analyze the effect of domain corpora and transfer learning techniques on ARC downstream tasks, in-domain corpora and close-domain corpora are developed in this work. To the best of our knowledge, this is the first publicly available domain corpora for transfer learning in the AEC domain.

(2) Based on the proposed domain corpora, three domain-specific word embedding models (i.e., in-domain, close-domain, and mix-domain model) and ten further pretrained BERT and ERNIE models are obtained.

(3) Six types of DL-based TC models and four types of DL-based NER models are deployed for the experiments. Subsequently, four experiments are conducted to reach the following conclusions. 1) There is no consistent global ranking of word embeddings for the TC and NER task. However, the domain-specific word embedding models can slightly prompt the performance of most deep models for TC and NER tasks in the ARC domain. 2) Further pretraining on an in-domain corpus is very useful for improving the performance of BERT for the TC task. For the BERT model, the larger the amount of in-domain corpora, the better the model performs. 3) The further pretrained BERT model is useful for improving the performance of the NER tasks in the ARC domain. 4) The contextual word embedding-based DL models perform better than the static word embedding-based DL models on both TC tasks and NER tasks.

(4) The BERT model further pretrained based on the proposed in-domain corpora achieves better performance than the original one for the typical downstream NLP tasks (i.e., TC and NER) in ARC. This study also open-sources the best models named ARCBERT-Large and ARCBERT-Small, which can help future related studies in ARC.

In the AEC domain, one challenge of DL models is the lack of labeled training data. The results show that the developed domain corpora, associated with transfer learning techniques, can enhance the performance of DL models without extra manual annotation. This can facilitate the development and further application of DL models in the ARC and AEC domains. These DL models can be used for TC and NER tasks, which are the two most vital tasks for the ARC domain. Typical further applications may include 1) automated rule interpretation by which researchers can filter irrelevant texts by TC to save time and improve accuracy for the subsequent tasks (Zhou & El-Gohary, 2016); 2) assigning more proper code generation functions to various kinds of regulatory sentences by identifying the features by TC, which aims to improve the coverage of automated rule interpretation (Solihin & Eastman, 2015); 3) using TC in construction site accident report classification (Cheng et al., 2020) and topic modeling (Lin et al., 2020); 4) using NER tasks to automatically extract information from texts to serve the automated interpretation application or to serve the recommender system for relevant rule clauses (Zhou et al., 2020; Moon et al., 2022); and 5) using NER for semiautomatic construction and extensions of knowledge graphs, which are currently mainly approached via manual methods (Leng et al., 2019).

Despite the success of our approaches, many future studies would be valuable for both academia and industry. For example, the performance of the proposed BERT model pretrained on in-domain corpora should be further validated on other datasets, and a more detailed parameter analysis on the influence of word embedding dimensions or model types (e.g., skip-gram model or CBOW) should be carried out in the future. Furthermore, the domain corpora can be further completed by containing more related texts to prompt the development of the AEC domain. In addition, despite the word embedding and further pretraining techniques analyzed in this work, more transfer learning and semi-supervised learning strategies should be explored for leveraging large-scale, pattern-rich general-domain corpora to solve downstream tasks in the AEC domain.

7 Conclusion

In this research, we systematically investigate how the domain corpus could enhance DL models for various NLP tasks in the AEC domain. First, both in-domain and close-domain corpora are developed and opened for further exploration and adoption. To the best of our knowledge, this is the first publicly available domain corpora for pretraining NLP models in the AEC domain. Then, based on various experiments, we illustrate the advantages of developed domain corpora and contextual word embedding-based DL models (e.g., BERT):

1) For the TC tasks, domain corpora can enhance both static word embedding-based DL models and contextual word embedding-based DL models, achieving 11.4% and 6.4% improvements in the weighted F1 score, respectively.

2) For the NER tasks, domain corpora can enhance both static word embedding-based DL models and contextual word embedding-based DL models, achieving 8.7% and 5.4% improvements in the macro F1 score, respectively.

3) For all the tested NLP tasks, the BERT model pretrained on domain corpora outperforms the static word embedding-based DL models, with maximum improvements of 8.1% in weighted F1 score for TC and 3.8% in macro F1 score for NER tasks, respectively.

4) Compared to the static word embedding-based DL models, contextual word embedding-based DL models (e.g., BERT and ERNIE) can capture the rich context-related information of a word or phrase

in a sentence; thus, their performance in NER tasks and TC tasks outperforms other models.

Finally, the pretrained BERT model called ARCBERT-Large is proposed and obtains a 94.4% overall weighted F1 score for the TC task. The pretrained BERT model call and ARCBERT-Small is proposed and obtains an 81.8% overall macro F1 score for the NER task. The proposed models achieves state-of-the-art performance on AEC downstream NLP tasks. The developed domain corpora and the further pretrained models demonstrate promising results in various NLP tasks and may shed light on various future investigations and applications in the AEC domain.

Acknowledgment

The authors are grateful for the financial support received from the National Natural Science Foundation of China (No. 51908323, No. 72091512), the National Key R&D Program (No. 2019YFE0112800), and the Tencent Foundation through the XPLORER PRIZE.

References:

[1] Wang, M., Wang, C. C., Sepasgozar, S., & Zlatanova, S. (2020). A systematic review of digital technology adoption in off-site construction: Current status and future direction towards industry 4.0. Buildings, 10(11), 204. https://doi.org/10.3390/buildings10110204

[2] Wu, C., Li, X., Guo, Y., Wang, J., Ren, Z., Wang, M., & Yang, Z. (2022a). Natural language processing for smart construction: Current status and future directions. Automation in Construction, 134, 104059. https://doi.org/10.1016/j.autcon.2021.104059

[3] Liao, W., Lu, X., Huang, Y., Zheng, Z., & Lin, Y. (2021). Automated structural design of shear wall residential buildings using generative adversarial networks. Automation in Construction, 132, 103931. https://doi.org/10.1016/j.autcon.2021.103931

[4] Hassan, F. U., Le, T., & Lv, X. (2021). Addressing Legal and Contractual Matters in Construction Using Natural Language Processing: A Critical Review. Journal of Construction Engineering and Management, 147(9), 03121004. https://doi.org/10.1061/(ASCE)CO.1943-7862.0002122

[5] Fuchs, S. (2021). Natural language processing for building code interpretation: systematic literature review report.

[6] Eastman, C., Lee, J. M., Jeong, Y. S., & Lee, J. K. (2009). Automatic rule-based checking of building designs. Automation in Construction, 18(8), 1011-1033. https://doi.org/10.1016/j.autcon.2009.07.002

[7] Ismail, A. S., Ali, K. N., & Iahad, N. A. (2017). A review on BIM-based automated code compliance checking system. In 2017 International Conference on Research and Innovation in Information Systems (ICRIIS), 1-6. https://doi.org/10.1109/ICRIIS.2017.8002486

[8] Zhou, P., & El-Gohary, N. (2016). Domain-specific hierarchical text classification for supporting automated environmental compliance checking. Journal of Computing in Civil Engineering, 30(4), 04015057. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000513

[9] Song, J., Kim, J., & Lee, J. K. (2018). NLP and deep learning-based analysis of building regulations to support an automated rule checking system. In ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction (Vol. 35, pp. 1-7). IAARC Publications. https://www.proquest.com/docview/2123611147?pq-origsite=gscholar&fromopenview=true

[10] Cheng, M. Y., Kusoemo, D., & Gosno, R. A. (2020). Text mining-based construction site accident classification using hybrid supervised machine learning. Automation in Construction, 118, 103265. https://doi.org/10.1016/j.autcon.2020.103265

[11] Zhang, R., & El-Gohary, N. (2021). A deep neural network-based method for deep information extraction using transfer learning strategies to support automated compliance checking. Automation in Construction, 132, 103834. https://doi.org/10.1016/j.autcon.2021.103834

[12] Zhou, Y. C., Zheng, Z., Lin J.R., Lu X.Z. (2020). Deep natural language processing-based rule transformation for automated regulatory compliance checking. Preprint. https://doi.org/10.13140/RG.2.2.22993.45921

[13] Moon, S., Lee, G., Chi, S., & Oh, H. (2021). Automated construction specification review with named entity recognition using natural language processing. Journal of Construction Engineering and Management, 147(1), 04020147. https://doi.org/10.1061/(ASCE)CO.1943-7862.0001953

[14] Leng, S., Hu, Z. Z., Luo, Z., Zhang, J. P., & Lin, J. R. (2019). Automatic MEP knowledge acquisition based on documents and natural language processing. In Proceedings of the 36rd CIB W78 conference. https://linjiarui.net/files/2019-09-18-automatic-mep-knowledge-acquisition-based-on-nlp.pdf

[15] Wu, L. T., Lin, J. R., Leng, S., Li, J. L., & Hu, Z. Z. (2022b). Rule-based information extraction for mechanical-electrical-plumbing-specific semantic web. Automation in Construction, 135, 104108. https://doi.org/10.1016/j.autcon.2021.104108

[16] Xu, X., & Cai, H. (2021). Ontology and rule-based natural language processing approach for interpreting textual

regulations on underground utility infrastructure. Advanced Engineering Informatics, 48, 101288. https://doi.org/10.1016/j.aei.2021.101288

[17] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[18] Fang, W., Luo, H., Xu, S., Love, P. E., Lu, Z., & Ye, C. (2020). Automated text classification of near-misses from safety reports: An improved deep learning approach. Advanced Engineering Informatics, 44, 101060. https://doi.org/10.1016/j.aei.2020.101060

[19] Mohan, S., Angell, R., Monath, N., & McCallum, A. (2021). Low Resource Recognition and Linking of Biomedical Concepts from a Large Ontology. arXiv preprint arXiv:2101.10587.

[20] Sun, Y., Wang, S., Li, Y., Feng, S., Chen, X., Zhang, H., Tian, X., Zhu, D., Tian, H., Wu, H. (2019). Ernie: Enhanced representation through knowledge integration. arXiv preprint arXiv:1904.09223.

[21] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539

[22] Manning, C., & Schutze, H. (1999). Foundations of statistical natural language processing. MIT press.

[23] Wu, C., Li, X., Guo, Y., Wang, J., Ren, Z., Wang, M., & Yang, Z. (2022). Natural language processing for smart construction: Current status and future directions. Automation in Construction, 134, 104059. https://doi.org/10.1016/j.autcon.2021.104059

[24] Li, H. (2017). Deep learning for natural language processing: advantages and challenges. National Science Review. https://doi.org/10.1093/nsr/nwx099

[25] Chen, Y. (2015). Convolutional neural network for sentence classification (Master's thesis, University of Waterloo). http://hdl.handle.net/10012/9592

[26] Liu, P., Qiu, X., & Huang, X. (2016). Recurrent neural network for text classification with multi-task learning. arXiv preprint arXiv:1605.05101.

[27] Lai, S., Xu, L., Liu, K., & Zhao, J. (2015, February). Recurrent convolutional neural networks for text classification. In Twenty-ninth AAAI conference on artificial intelligence.

[28] Johnson, R., & Zhang, T. (2017, July). Deep pyramid convolutional neural networks for text categorization. In Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 562-570).

[29] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

[30] Tian, D., Li, M., Shi, J., Shen, Y., & Han, S. (2021). On-site text classification and knowledge mining for large-scale projects construction by integrated intelligent approach. Advanced Engineering Informatics, 49, 101355. https://doi.org/10.1016/j.aei.2021.101355

[31] Zhong, B., Pan, X., Love, P. E., Ding, L., & Fang, W. (2020). Deep learning and network analysis: Classifying and visualizing accident narratives in construction. Automation in Construction, 113, 103089. https://doi.org/10.1016/j.autcon.2020.103089

[32] Li, R. Y. M., Li, H. C. Y., Tang, B., & Au, W. (2020, August). Fast AI classification for analyzing construction accidents claims. In Proceedings of the 2020 Artificial Intelligence and Complex Systems Conference (pp. 1-4). https://doi.org/10.1145/3407703.3407705

[33] Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space

odyssey. IEEE transactions on neural networks and learning systems, 28(10), 2222-2232. https://doi.org/10.1109/TNNLS.2016.2582924

[34] Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991.

[35] Ma, X., & Hovy, E. (2016). End-to-end sequence labeling via bi-directional lstm-cnns-crf. arXiv preprint arXiv:1603.01354.

[36] Zhong, B., Xing, X., Luo, H., Zhou, Q., Li, H., Rose, T., & Fang, W. (2020). Deep learning-based extraction of construction procedural constraints from construction regulations. Advanced Engineering Informatics, 43, 101003. https://doi.org/10.1016/j.aei.2019.101003

[37] Feng, D., & Chen, H. (2021). A small samples training framework for deep Learning-based automatic information extraction: Case study of construction accident news reports analysis. Advanced Engineering Informatics, 47, 101256. https://doi.org/10.1016/j.aei.2021.101256

[38] Moon, S., Lee, G., & Chi, S. (2022). Automated system for construction specification review using natural language processing. Advanced Engineering Informatics, 51, 101495. https://doi.org/10.1016/j.aei.2021.101495

[39] Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3(1), 1-40. https://doi.org/10.1186/s40537-016-0043-6

[40] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018, October). A survey on deep transfer learning. In International conference on artificial neural networks (pp. 270-279). Springer, Cham. https://doi.org/10.1007/978-3-030-01424-7_27

[41] Yao, Y., & Doretto, G. (2010, June). Boosting for transfer learning with multiple sources. In 2010 IEEE computer society conference on computer vision and pattern recognition (pp. 1855-1862). IEEE. https://doi.org/10.1109/CVPR.2010.5539857

[42] Pan, S. J., Tsang, I. W., Kwok, J. T., & Yang, Q. (2010). Domain adaptation via transfer component analysis. IEEE transactions on neural networks, 22(2), 199-210. https://doi.org/10.1109/TNN.2010.2091281

[43] Sogou. (2021). Sogou news. http://www.sogou.com/labs/resource/list_news.php (Access on 2021-12-11) (in Chinese)

[44] Wikipedia. (2021a). Wikimedia Downloads. https://dumps.wikimedia.org/ (Access on 2021-12-11)

[45] Myers, D., & McGuffee, J. W. (2015). Choosing scrapy. Journal of Computing Sciences in Colleges, 31(1), 83-89. https://www.researchgate.net/profile/James-Mcguffee/publication/314179276_Choosing_Scrapy/links/58b8a088a6fdcc2d14d9a326/Choosing-Scrapy.pdf

[46] Wikipedia. Category: Civil engineering (2021b) https://zh.wikipedia.org/wiki/Category:%E5%9C%9F%E6%9C%A8%E5%B7%A5%E7%A8%8B. (Access on 2021-12-11) (in Chinese)

[47] Encyclopedia of China Publishing House. (2009). Encyclopedia of China. https://h.bkzx.cn/ (Access on 2021-12-18) (in Chinese)

[48] Soujianzhu. Chinese Rules. https://www.soujianzhu.cn/default.aspx (accessed: June 22, 2021). (in Chinese)

[49] Wang, Y., Liu, S., Afzal, N., Rastegar-Mojarad, M., Wang, L., Shen, F., Kingsbury, P., Liu, H. (2018). A comparison of word embeddings for the biomedical natural language processing. Journal of biomedical informatics, 87, 12-20. https://doi.org/10.1016/j.jbi.2018.09.008

[50] Zhao, Z., Liu, T., Li, S., Li, B., & Du, X. (2017, September). Ngram2vec: Learning improved word representations from ngram co-occurrence statistics. In Proceedings of the 2017 conference on empirical methods in natural language processing (pp. 244-253). https://aclanthology.org/D17-1023.pdf

[51] Hugging Face. (2019). Bert-base-chinese. https://huggingface.co/bert-base-chinese/tree/main (Access on 2021-12-11)

[52] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013a). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

[53] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In Advances in neural information processing systems (pp. 3111-3119).

[54] Zheng, Z., Zhou, Y.C., Chen, K.Y., Lu, X.Z., Lin, J.R., She, Z.T. (2022). Text classification-based approach for automatically evaluating building codes' interpretability. (under review)

[55] Ethayarajh, K. (2019). How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. arXiv preprint arXiv:1909.00512.

[56] Solihin, W., & Eastman, C. (2015). Classification of rules for automated BIM rule checking development. Automation in construction, 53, 69-82. https://doi.org/10.1016/j.autcon.2015.03.003

[57] Lin, J. R., Hu, Z. Z., Li, J. L., & Chen, L. M. (2020). Understanding On-Site Inspection of Construction Projects Based on Keyword Extraction and Topic Modeling. IEEE Access, 8, 198503-198517. https://doi.org/10.1109/ACCESS.2020.3035214