Proceedings of the IISE Annual Conference & Expo 2023
K. Babski-Reeves, B. Eksioglu, D. Hampton, eds.

Deep Neural Network and Mixed-Integer Programming Framework for Scheduling Textual Construction Workplace Audits

Farouq Halawa, Szu-Kai Hayden, Raashid Mohammed
Global Engineering Technical Services
Amazon.com

Abstract

Construction auditing is a vital step in designing new sites in fulfillment centers (FCs). The audits include inspection processes for newly launched buildings which ensure that the buildings meet workplace standards. In current practice the process of scheduling construction audits and prioritizing the checklists when launching new buildings is highly manual, and requires going over lengthy textual data to determine dates of implementation and sequence. The lack of visibility on actual equipment installation dates result in conducting more auditing visits than needed and significant traveling and labor costs. Natural Language Processing (NLP) has been recently proposed to automate the auditing processes in construction sites. In this research, a hybrid machine learning and mathematical optimization framework is proposed to schedule and sequence the auditing checks. The proposed model bundles the checklists and assigns visit months based on historical probability of occurrence using Mixed Integer Linear Programming (MILP). A deep neural network with fine-tuned BERT-TSDAE architecture is proposed to generate embeddings that capture semantic similarities between checklist and checklist types and predict the occurrence time of the checks. The proposed model was validated on a subset of 49 warehouses, contributing to a potential reduction of 51.3 % of visits per site due to optimal assignment of visits. The proposed Sentence BERT fined-tuned with TSDAE model over performed CNN-Glove architecture, proposed in the literature for construction auditing, and led to the highest accuracy and F1 score.

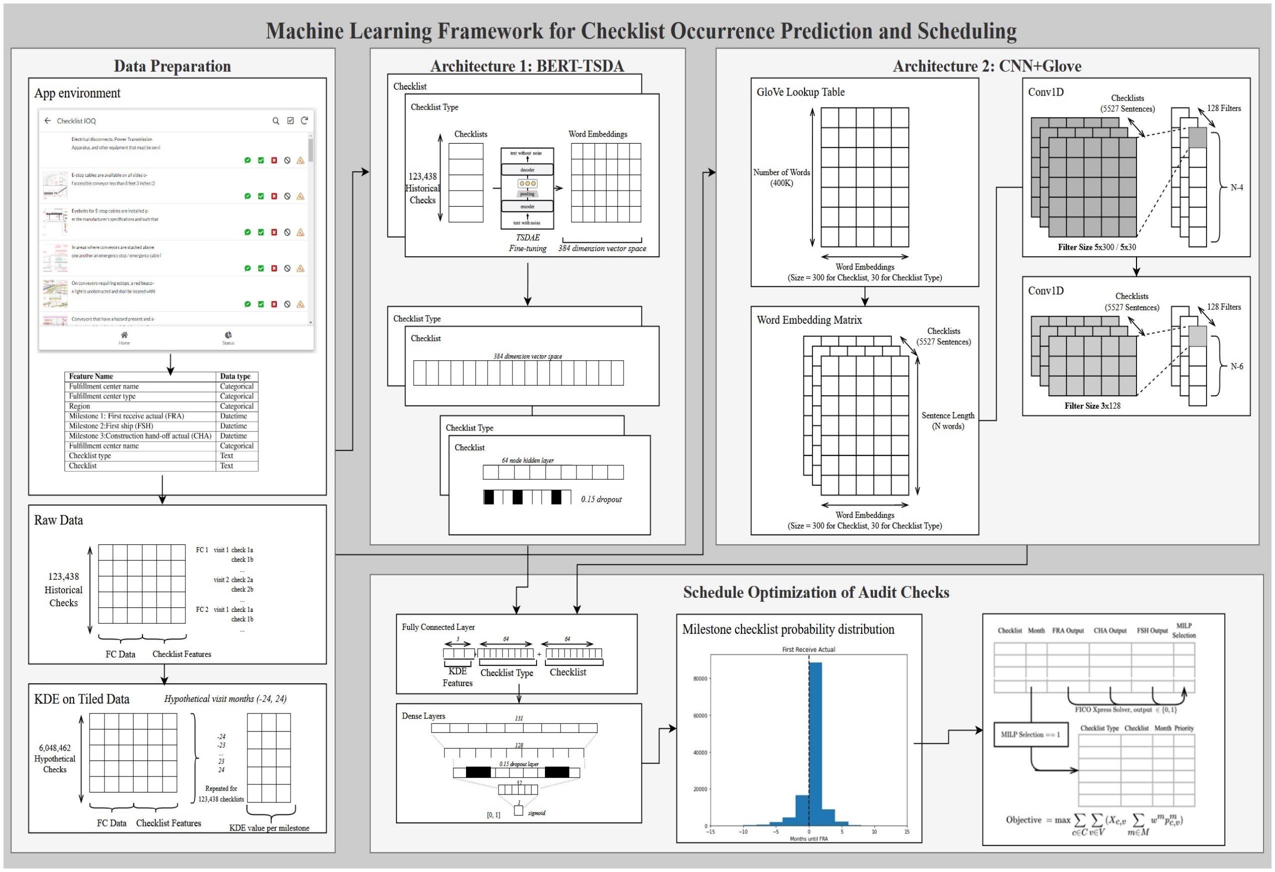
Keywords
Natural Language Processing, Deep Learning, Machine Learning, Data Science

1. Introduction

Construction inspection is a vital step to ensure that the new established buildings meet guidelines before the hand-over to operations. In this current practice, engineers and designers go over a large set of textual specifications in a manual fashion [1]. Lack of prioritization of checklists can create ambiguity and delays impacting commissioning timelines and targets. Natural language processing (NLP) has recently emerged as a promising tool to automate the construction inspection. NLP utilizes machine learning to extract semantic relationships and knowledge from textual datasets. The field of NLP in construction and design is still in its infancy due to limited algorithms and frameworks tested on such a dataset [2]. In the recent few years, several algorithms were proposed to automate the construction auditing [1–4, 9, 10], with multiple focuses including selection of relevant texts, clustering checklists, and duplicate detection. Nevertheless, the concept of checklist audit scheduling has not been tackled yet. Recognizing manually the relevant checklists in construction consumes at least one-third of an inspector time [2]. Significant time is also spent by firms to plan for the inspector visits, create schedules, and determine the optimal sequence of checks. Traditionally, the field of mathematical programming has been used to find optimal schedules of resources, staffing, or machinery [5, 7, 11]. In the context of NLP, sequencing textual data is a challenging task as the auditing checks are different depending on the type of site, type of asset, and the time-line of the construction plan. Often, the schedule is created based on the construction milestones. Thus, there are multiple factors that can influence the type of checks that are implemented. A fusion of machine learning and mathematical optimization is possible when having a large historical dataset of checklists at a warehousing network and a software to display the checks. With the advent of Industry 4.0, engineers can conduct those auditing checks using software applications. The integration of intelligent systems in those apps is significant as it enables the engineers to view the optimal lists of audits and select the ones that are relevant in an automated fashion. The proposed framework, can also be utilized in maintenance checks or any other application that is textual in nature. The contribution of this work is a novel deep learning & mixed-integer linear model to schedule checklist requirements and optimize the number of visits required by the designers and engineers, focusing on auditing requirements in the workplace. The remainder of this paper details the research methodology in Section 2, provides experimental results in Section 3, and indicates the research conclusion and future works in Section 4.

2. Methodology

The methodology of scheduling audits and prioritizing checklists is divided into four main phases, including data preparation, feature generation, checklist selection, and finally scheduling. Data preparation is performed to combine the checklist data for all sites from a software app environment, and transform the dataset into a format that also represents the checklist occurrence over time. Feature generation is performed through Kernel Density Estimation and word embeddings, followed by a deep neural network to predict checklist occurrence, and a mixed-integer linear programming model to assign checklists to visits. Figure 1 provides a high level illustration of the proposed framework. Detailed explanation of the methodology is shown in sections 2.1-2.4.


Machine Learning Framework for Checklist Occurrence Prediction and Scheduling
Data Preparation
App environment
Raw Data
123,438 Historical Checks
FC Data
Checklist Features
KDE on Tiled Data
Hypothetical visit months (-24, 24)
6,048,462 Hypothetical Checks
Repeated for 123,438 checklists
KDE value per milestone
Architecture 1: BERT-TSDA
Checklist
Checklist Type
Checklists
Word Embeddings
TSDAE Fine-tuning
384 dimension vector space
64 node hidden layer
0.15 dropout
Architecture 2: CNN+Glove
GloVe Lookup Table
Number of Words (400K)
Word Embeddings (Size = 300 for Checklist, 30 for Checklist Type)
Word Embedding Matrix
Checklists (5527 Sentences)
Sentence Length (N words)
Conv1D
128 Filters
Filter Size 5x300 / 5x30
Filter Size 3x128
Schedule Optimization of Audit Checks
Fully Connected Layer
KDE Features
Checklist Type
Checklist
Dense Layers
0.15 dropout layer
sigmoid
Milestone checklist probability distribution
First Receive Actual
Months until FRA
FICO Xpress Solver, output ∈ {0,1}
MILP Selection == 1
Objective = max $\sum_{c\in C}\sum_{v\in V}(X_{c,v}\sum_{m\in M} w^m p^m_{c,v})$

Feature Name	Data type
Fulfillment center name	Categorical
Fulfillment center type	Categorical
Region	Categorical
Milestone 1: First receive actual (FRA)	Datetime
Milestone 2:First ship (FSH)	Datetime
Milestone 3:Construction hand-off actual (CHA)	Datetime
Fulfillment center name	Categorical
Checklist type	Text
Checklist	Text

Figure 1: Proposed hybrid ML and MILP framework for scheduling checklist visits, showing the vectors representation and sequence of steps

2.1 Data Preparation

Two main types of data are used in the framework, fulfillment center data and checklist data. Fulfillment center data contains several construction milestones, assumed in this work as construction actual time (CHA) (i.e., time when the construction ends), facility first-received actual time (FRA) (i.e., time when the facility receives first items to store), facility first ship actual time (FSH) (i.e., time when the facility starts shipping items and packages). The number of milestones can vary depending on the site, however, those are the basic ones in warehouses. Checklist data includes a list of all textual auditing checks that were historically performed at a site or fullfilment center (FC). The model features are outlined in Figure 1, after being extracted from a live software application. The dataframes are merged on fulfillment center name to produce a dataset containing both fulfillment center and checklist level features. Each checklist has a specific datetime at which it was performed. The difference between each milestone date and the checklist historical date is then calculated and is expressed in number of months. To avoid over-fitting, the dataframe was tiled by duplicating the features for all available months in a time-line of 2 years. Figure 1 further visualizes the data structure discussed in this phase. The model labels are encoded as "1", for the month in which the check was performed historically, and "0" on all other months. As a result, the dataset to train the model will be highly imbalanced since the non-occurrences will exceed the occurrences. The model was trained on a dataset of 6,048,462 data points, which is highly imbalanced with 97.96% of rows representing checklist non-occurrences and only 2.04% of rows representing checklist occurrences. The number of FCs is randomly split into train, test, and validation sets of size 316, 25, and 24. The split resulted in datasets of length 5,153,085, length 463,932, and length 431,445, respectively.The existing dataframe includes a combination of features with different types: categorical, text, and time specific. The feature generation part aims to transform the text features into continuous values and to extract additional features that supports the prediction of checklist occurrence, as explained in the coming sections.

2.2 Feature Generation Using KDE

Checklist occurrence counts in relation to each milestone take on discrete values which may not reflect the true probability of occurrence. For example, if a checklist was historically only performed within one month of the FRA milestone, using discrete counts would imply that a checklist has a 0% probability of occurrence in any other month other than the FRA month. To tackle this, KDE (Kernel density estimation) [6] was implemented to use historical data to generate probability of checklist occurrence over an interval of two years before and after each milestone. Let n be the number of data points, σ is the standard deviation or bandwidth of the Gaussian distribution, x_i is the value of a particular data point i and $\hat{x}$ is the function input or the value that the kernel density is evaluated. Equation (1) represents the KDE function. KDE was performed 3 times for each of the milestones.

$$f(\hat{x}) = \frac{1}{n\sigma\sqrt{2\pi}} \sum_{i=0}^{n} e^{-\frac{1}{2}(\frac{x_i-\hat{x}}{\sigma})^2} \tag{1}$$

2.3 Checklist Occurrence Prediction

Two state-of-the-art architectures were tested, and the results produced by training each architecture on the 3 milestones were used to select the best model. The purpose of each neural network is to output the probability of the two classes [0,1], as a proxy for the probability of occurrence of a given checklist at a given number of months away from a milestone. For each model, each individual checklist will have 48 predictions indicating the probability of occurrence from 24 months before to 24 months after each milestone.

BERT & TSDAE: In this work, we implement the framework proposed by Wang et al. 2021 [8] for text embeddings. BERT allows for the production of fixed size vectors of sentence embeddings given variable length input sentences. A pre-trained all-MiniLM-L6-V2 BERT model was initialized, and fine-tuned over 18 epochs of training using TSDAE (Transformers and Sequential Denoising Auto-Encoder) with a dataset consisting of 5,290 unique checklists and checklist types with a word length greater than 10. During the training process, the sentences are damaged through the removal of words and then encoded into fixed-size vectors, while the model attempts to reconstruct the vectors into the original, undamaged inputs. The embeddings are then passed into 3 neural networks used in this work, each consisting of an input layer receiving two sets of size 384 sentence embeddings, followed by two size 64 hidden layers. These hidden states are then concatenated with each other and the 3 milestone-specific KDE features, followed by 3 hidden layers of size 128, 128, 32 and output a single value between 0 and 1 using a sigmoid activation function, as they aim to predict the likelihood of checklist occurrence.Both models were trained using a binary cross-entropy loss function.
Glove & CNN: This architecture was proposed by Jeon et al. 2021 [2] for checklist auditing applications. In this work, we adapted the architecture as the following. The network consists of a size 300 embedding layer for a checklist and a size 30 embedding layer for checklist type, which each take a vectorized representation of words, followed by a convolutional layer that scans across the word embeddings to output a hidden layer of sentence embeddings. These hidden states are then concatenated with each other and the 3 milestone-specific KDE features, followed by 3 hidden layers of size 128, 128, 32. For the two tested architectures, the loss function is represented in equation 2 using Binary Cross-Entropy, where y_i is the target label of the data sample, $\hat{y}$ is the model output and n is the number of samples in

the training dataset (equation 2). Due to the imbalance in the data caused by having more months where the check is not implemented (being labeled as 0), we are giving more importance to the minority class. Given a set of classes C, the observations of each class is x_i, and the weight of a class is shown in equation 3.

$$\text{Loss} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \cdot \log\left(\hat{y}_i\right) + \left(1-y_i\right)\cdot \log\left(1-\hat{y}_i\right)\right] \tag{2}$$

$$W_i = \frac{1}{|x_i|} * \frac{\sum_{i\in C}|x_i|}{2} \tag{3}$$

2.4 Mixed Integer Linear Program (MILP) for Visits Optimization

Given the results of section 2.3, the MILP model bundles the checklists and schedules them into months. Let $i \in 1,2,..$ represents the number of checks, j $\in 1,2,..$ represents the number of visits, and $m \in 1,2,..$ represents the construction milestones. The objective function is to assign each checklist to a specific visit so that the total assignment probability coming from ML selection model is maximized for a construction milestone $p^m_{i,j}$. The variable $x_{i,j}$ is a binary variable that is 1 if the check is assigned to a visit j, and w^m is a weight given to each milestone, as shown in equation (4).

$$\text{Objective } = \max \sum_{i=1}\sum_{j=1} X_{i,j} \sum_{m=1} w^m p^m_{i,j} \tag{4}$$

The model is subject to multiple constraints. Equation 5, makes sure that each check will be assigned to a visit, and thus all checks will be assigned.

$$\sum_{j=1} X_{i,j} = 1, \forall i \in N \tag{5}$$

Let T_i be the time it takes for one check to be performed. Let J_j be the available hours per visit j. The following equation state that the total number of checks in a visit should not exceed the available time, as shown in equation 6.

$$\sum_{i=1} X_{i,j} \leq T_i J_j, \forall j \in N \tag{6}$$

To avoid the case that we have one visit with only few checks, let us define V_j as a semi-continuous variable, where l is the lower threshold, explained in equation 7.

$$V_j \geq l \text{ or } V_j = 0, \forall j \in N \tag{7}$$

We then set the sum of checks per visit equal to V_j to ensure that each visit is assigned to a pre-defined number, as explained in equation 8.

$$\sum_{i=1} X_{i,j} = V_j, \forall j \in N \tag{8}$$

3. Experimental Results

This section examines the results of implementing the proposed models on a subset of 25 fulfillment centers.

3.1 Checklist Occurrence Model Selection

Two model architectures were tested in Table 1 on three construction milestones.

Table 1: Selection of Machine learning architecture using different accuracy metrics

Milestone	GloVe with Convolutional Neural Network						Sentence BERT & TSDAE Deep Neural Network					
	Accuracy	Precision	Recall	F1	ROC AUC	Time/epoch	Accuracy	Precision	Recall	F1	ROC AUC	Time/epoch
FRA	0.96	0.97	0.96	0.97	0.95	674s	0.97	0.98	0.98	0.98	0.95	22s
CHA	0.96	0.96	0.95	0.96	0.95	680s	0.97	0.98	0.97	0.97	0.96	23s
FSH	0.97	0.98	0.95	0.97	0.94	675s	0.97	0.98	0.98	0.98	0.96	23s
Average	0.96	0.97	0.95	0.97	0.95	676s	0.97	0.98	0.98	0.98	0.96	23s

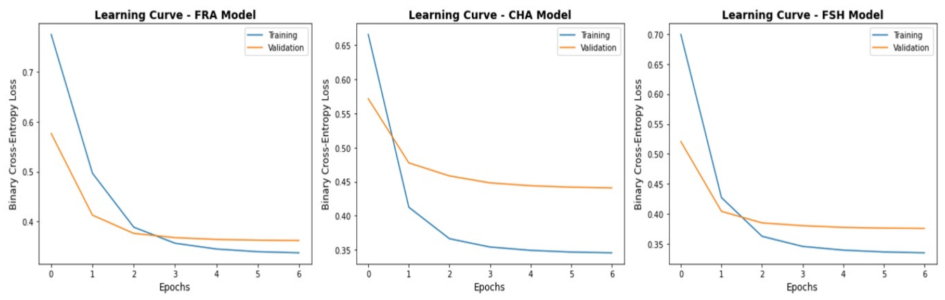


Figure 2: Model learning curves using 3 construction milestones (FRA, CHA, FSH)

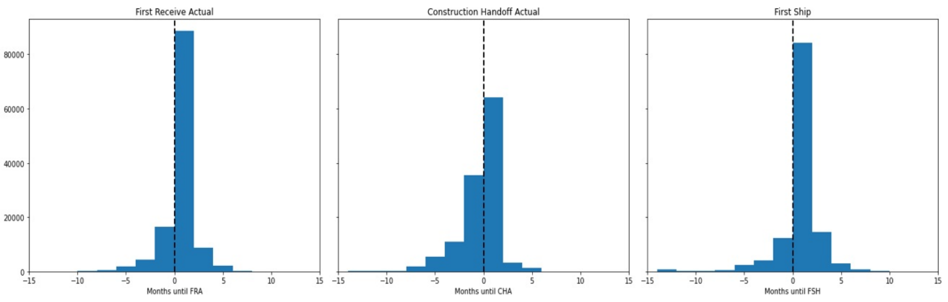


Figure 3: Distribution of checklists using 3 construction milestones (FRA, CHA, FSH)

As shown in the comparison tables, the BERT & TSDAE Deep Neural Network model was selected for use in the final implementation, based on a slightly improved performance and a significantly faster run time over the GloVe & CNN model. The loss curves for the BERT & TSDAE deep neural network are represented in Figure 2. Figure 3 visualizes the distribution of all checklists across a time-line of 2 years using one of the construction milestones (FRA) as obtained from the model. The classification probabilities resulted from the machine learning models were used to create the histogram shown in Figure 2. However, this is not enough as the current results would not provide an estimation on how many visits are required, which is the focus of the MILP phase.

3.2 Mixed-Integer Linear Programming (MILP) Model Results

MILP model was applied to 49 FC's. Number of visits ranged between 1 to 11 . Overall, we find that the average number of visits per fulfillment center was reduced by 51.3%, shown in Figure 4 for a sample of 10 sites. Figure 4 shows that the model is more beneficial for sites that included high number of visits (>5 visits).

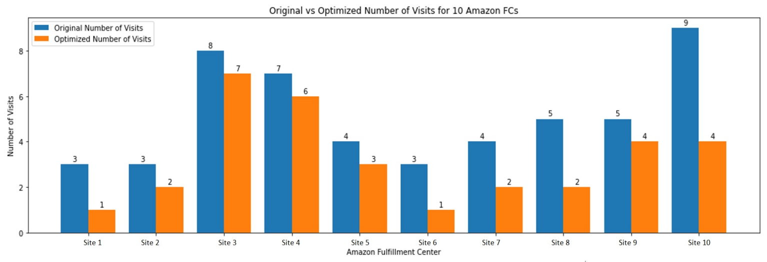


Figure 4: Comparison of number of auditing visits at 10 sites before and after implementing the proposed algorithm

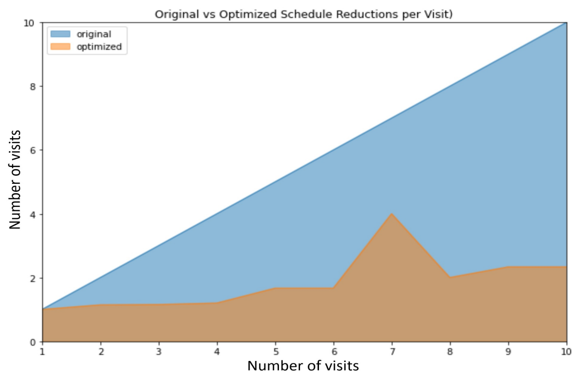


Figure 5: Reduction rate in number of visits: Simulation of 49 sites in 2022

4. Conclusion and Future Works

Optimal scheduling of checklists is a crucial step in construction and can increase the efficiency and reduce the number of auditing visits in the auditing process while ensure all facilities meet the installation and operational standards and specifications. The existing process of scheduling audits and prioritizing checklists is highly manual, leading to a high monetary cost from extra visits, as well as many other intangible costs and wasted time. This study demonstrated the value of applying the proposed novel machine learning and mathematical optimization approach to generate schedules of checklists for each fulfillment center visit, which can result in significant cost avoidance and improvement of workforce efficiency and work life harmony without adversely impacting existing operations. The proposed framework does not currently consider the time taken to complete checklists, and uses a naïve approach of setting a maximum number of checks per visit. Future work should consider the time aspect in the model. Future work should also consider other machine learning architectures to handle imbalanced data.

References

[1] Halawa, F., Abdul, M., Mohammed, R., 2022. "Applying Machine Learning for Duplicate Detection, Throttling and Prioritization of Equipment Commissioning Audits at Fulfillment Network". IISE EXPO Proceedings.

[2] Jeon, J., Xu, X., Zhang, Y., Yang, L., Cai, H., 2021. "Extraction of Construction Quality Requirements from Textual Specifications via Natural Language Processing". Transportation Research Record, 2675(9), 222-237.

[3] Yuan, C., Park, J., Xu, X., Cai, H., Abraham, D. M., Bowman, M. D., 2018. "Risk-based Prioritization of Construction Inspection. Transportation Research Record", 2672(26), 96-105.

[4] Goh, Y. M., Ubeynarayana, C. U., 2017, " Construction Accident Narrative Classification: An Evaluation of Text Mining Techniques. Accident Analysis Prevention, 108, 122-130.

[5] Hou, G. F. C., Thekumparampil, K., Oh, S., 2021. Multistage Stepsize Schedule in Federated Learning: Bridging Theory and Practice. In ICML Workshop (Vol. 12).

[6] Fan, J, James S., 1994. “Fast Implementations of Nonparametric Curve Estimators”. Journal of Computational and Graphical Statistics 3(1)

[7] Rohaninejad, M., Janota, M., Hanzálek, Z. (2023). "Integrated Lot-sizing and Scheduling: Mitigation of uncertainty in Demand and Processing Time by Machine Learning". Engineering Applications of Artificial Intelligence, 118, 105676.

[8] Wang, K., Reimers, N., Gurevych, I., 2021. "Tsdae: Using Transformer-based Sequential Denoising Auto-encoder for Unsupervised Sentence Embedding Learning". arXiv preprint arXiv:2104.06979.

[9] Chokor, A., N, H., Chong, W. K., El Asmar, M., 2016. "Analyzing Arizona OSHA Injury Reports Using Unsupervised Machine Learning". Procedia engineering, 145, 1588-1593.

[10] Zhang, J., El-Gohary, N. M., 2016. "Semantic NLP-based Information Extraction From Construction Regulatory Documents for Automated Compliance Checking". Journal of Computing in Civil Engineering, 30(2), 04015014.

[11] Erekat, A., Sweidan, H., Mdathil, S. C., Khasawneh, M., 2019. "Staffing and Scheduling Framework of Merged IR and CathLab Departments". IISE EXPO Proceedings, 360-365.