Automation in Construction 132 (2021) 103929

Contents lists available at ScienceDirect

Automation in Construction

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

Transformer machine learning language model for auto-alignment of long-term and short-term plans in construction

Fouad Amer [a,*], Yoonhwa Jung [a], Mani Golparvar-Fard [b]

[a] Department of Civil and Environmental Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
[b] Civil Engineering, Computer Science and Tech Entrepreneurship, University of Illinois at Urbana-Champaign, USA

ARTICLE INFO

Keywords:
Construction planning
Project controls
Machine learning
Natural language processing
Artificial intelligence

ABSTRACT

In construction, master schedules and look-ahead plans are created at different times (monthly vs. weekly), by different personas (planner vs. superintendent), with different software (scheduling solution vs. spreadsheet), and at different levels of granularity (milestones vs. production details). Their full-alignment is essential for project coordination, progress updating, and payment application reviews, and its absence may lead to costly litigation. This paper presents the first attempt to automate linking look-ahead planning tasks to master-schedule activities following an NLP-based multi-stage ranking formulation. Our model employs distance-based matching for candidate generation and a Transformer architecture for final matching.[1] Validation results from real-world projects demonstrate that the method helps planners match look-ahead planning tasks to master schedule activities by presenting a list of top-five matches with a precision of 76.5%. We also show that the method helps superintendents create look-ahead plans from a master schedule by generating lists of tasks based on activity descriptions.

1. Introduction

One of the challenges in managing field operations is reconciling project's master schedule (*long-term plan*) with superintendents' look-ahead plans (*short-term plans*) [1]. For each construction project, a master schedule is issued at the beginning of the construction phase to outline milestones and work activities from beginning to the end. These master schedules serve many purposes ranging from informing owners on state of progress, establishing long-term coordination among crews and trade contractors, to specifying terms of payment [2]. However, these initial schedules often do not accurately capture details too far into the future because information about actual duration, specific workflows, and deliveries are typically not available at an early stage of a project [3,4]. To reliably coordinate and direct various trades and crews working on the job, superintendents create and use short-term schedules, often called "look-ahead plans" [5].

The look-ahead plans outline which crews and trades should be at each specific location and what they should be doing at certain times [6]. In addition to detailed tasks that directly correspond to activities from the master schedule, these look-ahead plans may also document work not included in the master schedule. Thus, in the Last Planner System [3,7,8], look-ahead plans are often considered as the missing link in production control. Often, *issue management* systems or *task constraints* in the Last Planner System are also used to capture and track those field issues that are not represented in the master schedule and/or look-ahead plan. During coordination meetings, superintendents communicate, explain and revise these look-ahead plans with crews and trade contractors [9]. Superintendents use these look-ahead plans to coordinate work through daily foremen meetings, find out how the job is progressing, and what problems have arisen.

To ensure project managers, executives and owners are able to make sense of what is going on in the field, it is critical to tie in the day-to-day look-ahead planning tasks and issues (i.e., tasks in an issue management system or the *task constraints* in the Last Planner System) with the master schedule activities [10]. This is even a greater challenge in lean construction projects where creating weekly work/ look-ahead plans and then updating master schedule based on these short-term plans is common practice [11]. *Bridging this gap between master schedule and look-ahead plans to get the big picture has been a constant struggle* [1].

This problem is exacerbated when progress reporting is considered [12]. The master schedule and actual information about progress reported against look-ahead plans are also disconnected. Master schedules

* Corresponding author.
E-mail addresses: famer2@illinois.edu (F. Amer), yoonhwa2@illinois.edu (Y. Jung), mgolpar@illinois.edu (M. Golparvar-Fard).

[1] Access to the code and a sample of the dataset can be granted upon direct request from the authors.

https://doi.org/10.1016/j.autcon.2021.103929
Received 18 March 2021; Received in revised form 23 August 2021; Accepted 27 August 2021
Available online 16 September 2021

reside in standard construction planning solutions (e.g., Oracle Primavera, Microsoft Project), while look-ahead plans containing progress information live in spreadsheets or are printed in post-it notes on the trailer walls (See Fig. 1). Project managers need to be constantly aware of all activities on their sites to make sure their projects are on schedule, however, the disconnection between the short-term and long-term planning systems renders updating the master schedule activities from weekly tasks a challenging and time consuming endeavor.

The task of reporting progress is also often relayed to subcontractors or to most junior engineers on a project. It is a constant struggle making phone calls and chasing subcontractors to get progress information, compiling weekly progress reports, or typing up progress information into project schedules. Hence, in many circumstances, look-ahead plans are not progressed on a regular basis. Because look-ahead tasks and their progress status are disconnected from the master schedule activities, superintendents, engineers, and project managers are not aligned on the big picture. This back and forth that happens in the field delays the receipt of actual progress information by executives, often leaving them with poor visibility into the real state of progress on their projects.

These challenges necessitate project-level managers, superintendents, schedulers and often regional executives to get together on a monthly basis in costly meetings to discuss how much progress is made [12], explain why certain work packages are behind schedule, review planning alternatives, and update master schedules with progress data for monthly owner reporting and managing payment applications [13]. The nature of updating progress information retroactively into the master schedule leads the executive team not to consider look-ahead plans or progress reported by the field team. Instead, they rely on experience and gut feelings to manually update master schedules, as opposed to using real data from the field.

To eliminate the labor-intensive nature of manually aligning master schedules and look-ahead plans, a method is presented that automatically maps master schedule *activities* to look-ahead planning *tasks*. This terminology of *activities* and *tasks* is used throughout the manuscript to differentiate between master schedules and look-ahead plans. Our natural Language Processing (NLP)-based method uses a state-of-the-art Transformer, namely GPT-2 [14], to automatically map activities and tasks to one another. To validate the method, we obtained data from four construction projects and we manually linked tasks to their relevant activities. We then trained and tested different models and compared their ability in ranking activities based on their relevance to input tasks. The following sections review the literature related to matching tasks to activities, present our method, and discuss the results, limitations, and potential future expansions.

2. Literature review

One of the core principles in lean construction is to improve workflow reliability by continuously aligning *what will be done* on projects with *what should be done* through collaborative planning and a systematic application of the *Make Ready* Process [10,4,15]. While there is tremendous evidence that the Last Planner System [16] improves collaboration and reduces variability in near-term work execution [11, 17,4,15], its impact on master schedules has been a difficult subject for systematic analysis. A key reason is that long-term and short-term planning practices are often performed separately [1]. Because aligning short-term and long-term plans requires significant effort, during work execution, most project teams do not maintain their remaining work in alignment with the master schedule targets [10]. Fig. 2 shows two examples of these mis-alignments after importing and manually aligning look-ahead planning tasks with their respective master schedule activities. These issues are not specific to lean construction projects, as they happen similarly to conventional projects between master schedule activities and superintendent look-ahead plans.

While tracking performance of both long-term and short-term plans is essential to improving the productivity and efficiency of the project for future planning optimization [18], keeping all plans at every level of detail in alignment proves to be a challenging and time-consuming task [1]. Different scheduling software packages are used to prepare long-term and short-term schedules such as Oracle P6 and Microsoft Project for master schedules on one hand and Excel/spreadsheets for look-ahead planning and weekly work plans (WWP) on the other. This creates an epistemic bias when developing look-ahead plans due to poor linkage between production schedules and project progress milestones [5]. When weekly work plans or look-ahead plans are not properly linked to long term plans, percent plan complete (PPC) becomes loosely linked to project progress [5]. Also, the existing alignment method is labor-intensive where a project engineer or a superintendent has to manually update the progress of master schedule activities based on the progress reported on the relevant weekly tasks from the look-ahead plans. To date, formalizing the problem of aligning master schedule activities to look-ahead planning tasks and automating the process has received little to no attention. While there are available commercial solutions that facilitate the creation of aligned long-term and short-term plans, they rely on the human planner to manually perform the alignment by adding look-ahead planning tasks within the same Work Breakdown Structure (WBS) of the master schedule. Similarly, existing metrics that help monitoring the alignment such as Percent Required Completed or Ongoing (PRCO) [10] assume the alignment is already established.

The challenge in automatically aligning tasks to activities lies in their representation. Both activities and tasks are described using Natural Language – English – expressions with little or no standardization, grammatical errors, and project and construction-specific terms and abbreviations [19]. Researchers have already proposed methods for formalized representation of construction activities [20–22]. Amer et al. [23] offers an exhaustive literature review on the entire body of AI-driven methods in construction planning and scheduling domain. The latest method is Amer and Golparvar-Fard [24] where an NLP-driven pipeline is presented for deciphering construction activity descriptions and automatically decomposing them into their different constituents, known as the ALOR set: ⟨*Action, Location, Object, Responsible Party*⟩. Building on the ALOR method of decomposing schedule activity descriptions, Amer and Golparvar-Fard [19] formalized a new

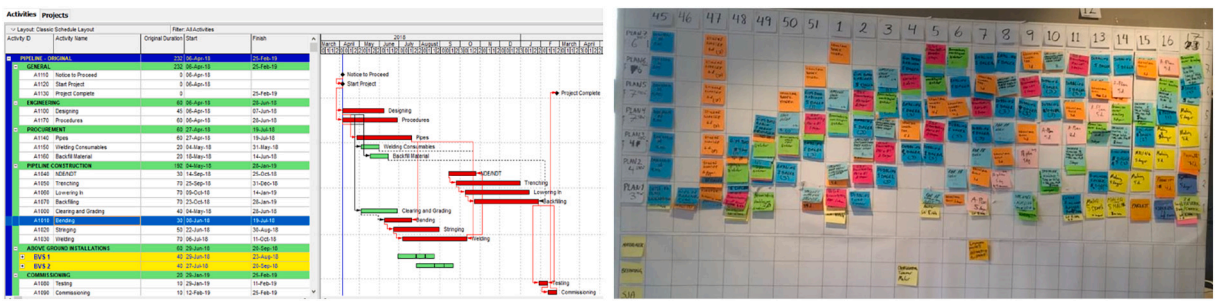

Fig. 1. A project manager's master schedule vs. a superintendent's look-ahead plan.

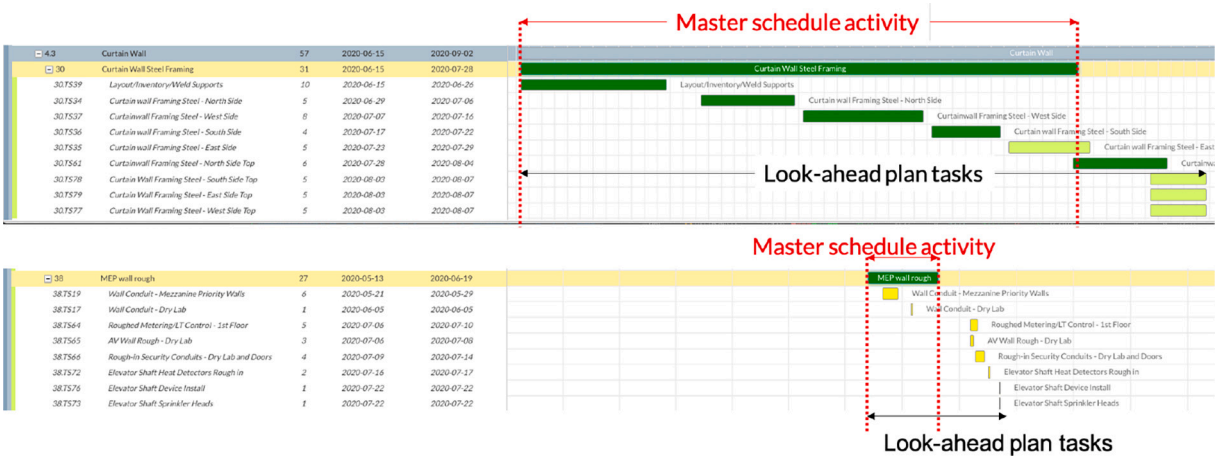

Fig. 2. Two examples of mis-alignments between master schedule activities and their corresponding look-ahead planning tasks (activities and tasks are shown with dark and light green colors respectively). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

representation of construction activities and leveraged Long Short-Term Memory Recurrent Neural Networks (LSTMs) to model and learn construction sequencing knowledge from existing project records. Similarly, Alikhani et al. [25] used Bidirectional LSTMs to learn activity sequences. Zhao et al. [26] applied NLP techniques to extract construction methods and dependency logic from construction schedules. Their method is based on lookup tables and *n*-gram models to automatically detect and rectify errors in project schedule activity descriptions, and Part-of-Speech (POS) tagging to identify nouns and verbs as part of a pipeline for identifying construction methods. Therefore, from a research standpoint, the knowledge gaps are (1) the lack of a formalization of the problem of automatically matching short-term and long-term plans as well as (2) the absence of methods and systems that leverage NLP to enable and facilitate that matching.

In addition to their use for analyzing and automating scheduling related tasks, NLP techniques have been widely applied to information acquisition and retrieval in the construction industry [27]. In particular, NLP techniques such as POS tagging were utilized to recognize the syntactic and semantic features presented in an input document, such as a construction contract, for information extraction purposes and to support automated compliance checking. For instance, Zhang et al. [28] used NLP techniques to automatically check and extract project information and requirements from textual regulatory documents and Building Information Models (BIM). Also, Zhou et al. [29] presented an NLP-based model to automatically check the compliance between BIM-represented building designs and environmental requirements in contract specifications. Similarly, NLP techniques such as tokenization, parsing, and POS tagging are applied to facilitate the ordering of Requests for Information (RFIs) where keywords are extracted and mapped to Industry Foundation Classes (IFC) entities and attributes [30].

The alignment of short-term to long-term plans lies at the heart of construction progress monitoring. In the recent years, many researchers focused on automating the process of progress detection and relaying it back to the construction schedule. To detect progress, researchers resorted to Photogrammetry and computer vision techniques to generate 3D point clouds from images, videos, or laser scanners to capture site conditions for outdoor and indoor construction [31,32]. Given a collection of unstructured site images (2D or 3D images), Structure-from-Motion (SFM) and Multi-View Stereo (MVS) techniques are used to reconstruct the scene in 3D with high fidelity. Similarly, Simultaneous Localization and Mapping (SLAM) is used to recreate 3D scenes from videos (structured and consecutive images) [12]. After the actual site conditions are captured in 3D, the point clouds are aligned and compared to BIM to detect progress. Similarly, [33] investigated the use of RFID tags to track and control material deliveries and supply chains, and [34] presented a method for detecting reporting requirements from construction contracts using NLP.

To the best of our knowledge, this paper presents the first attempt towards the automatic alignment of long-term and short-term plans, particularly using NLP techniques. We formalize the problem, and we present a data-driven model that learns the dependencies between long-term and short-term plans, purely based on schedule activity and look-ahead planning tasks descriptions. The alignment knowledge learned by the model is also employed to generate a list of look-ahead planning tasks based on an input master schedule activity turning the model into a generative tool that assists planning efforts and augments the knowledge of planning engineers and superintendents. The next sections discuss the method in detail, present the experimental setup and results, and draw conclusions and recommendations for future research.

3. Method

3.1. Problem definition

Master schedule activities and look-ahead planning tasks are created and organized in different formats for different purposes. In most cases, tasks in the look-ahead plans are created in spreadsheets with minimal to no WBS information. On the other hand, activities are created in commonly used master scheduling solutions such as Oracle P6 or Microsoft Project, and they are organized with reliable WBS information. While activities are developed by project planners, tasks are created by superintendents. Tasks are more granular and activities are more general. Because of these differences in how they are created and organized, the language used to describe activity vs. task description is different, and it is challenging for an automated system to align them.

We present a scalable and automated method for identifying the relevant master activity for an input look-ahead planning task (See Fig. 3). We formulate the problem as a ranking problem where the master activities are ranked in order of their relevance to the input look-ahead planning task. Mathematically, given a look-ahead planning task w_i and a set of master schedule activities $A = \{a_1, a_2 \dots a_j\}$, the goal is to rank A in the order of relevance to w_i and to retrieve (a) the single correct master activity match or (b) the top five matches to facilitate the workflows of a human planner. Both task w_i and activity a_j are represented using natural language expressions with little or no standardization.

We approach the problem from multiple angles: First, we use unsupervised distance-based matching to establish a baseline for comparison and to highlight the differences between task and activity representations. Second, we extract semantic location information from both look-

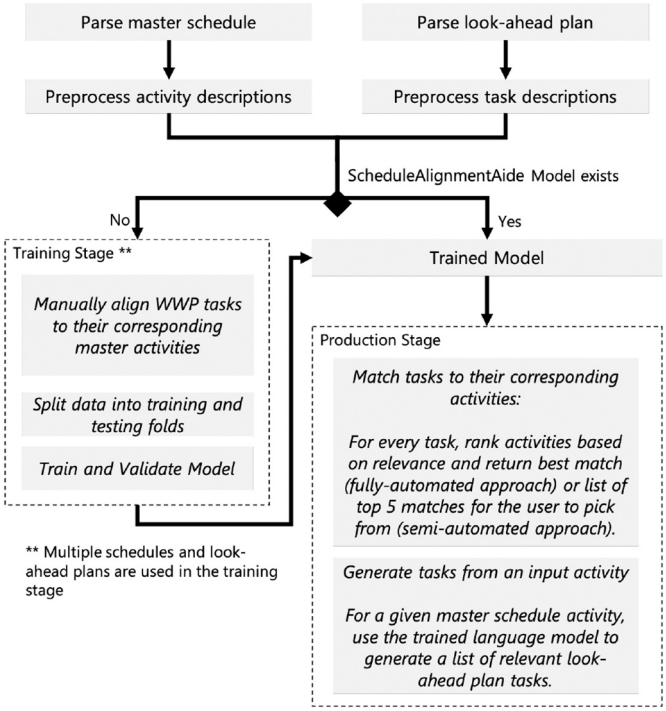

Fig. 3. Workflow of the proposed model at training and production times.

ahead planning tasks and master activities and we match locations to assist distance-based matching. Third, we introduce a supervised machine learning model built on the state-of-the art Transformer-based neural network GPT-2 [14], and we use it to improve matching results in a multi-stage ranking setup [12]. On the other hand, we explicitly stay away from using any temporal matching, i.e., matching based on the start and end dates of tasks and activities. The reason for that is the large variance of delays among different projects and among different activities within the same project. The dataset we obtained shows tasks linked to activities which were scheduled to start more than one month before the task's start date as well as tasks linked to activities which are scheduled to start in the future. Therefore, we wanted our approach to be completely NLP driven and independent of the delays specific to each input schedule.

3.2. Main contribution and model use case

This model is to be used to align short-term plans (look-ahead plans and WWPs created by superintendents –during pull planning sessions in projects following lean practices–) to their corresponding long-term plans (master schedules created by project planners) for reporting purposes. We envision a tool based on this model to be used by the project site team as follows:

"*The project engineer or the project planner would start by uploading the master schedule to the tool. Then, whenever an update is required, for example monthly, the superintendent will upload their look-ahead plans. The tool will then use AI to establish matches between the tasks listed on the look-ahead plans and the master schedule activities. Based on the progress reported for each task, the tool will update the progress of the relevant master activity and request the feedback and approval of the superintendent. The tool will then create a report detailing the overall site progress.*"

Reaching the above level of automation requires:

- Establishing the links between the uploaded look-ahead plan tasks and the relevant master schedule activities (focus of this manuscript).
- Assessing the level of contribution of the look-ahead plan task to its master relevant master schedule activity. Activities are often more general and have large scope than their corresponding look-ahead plan tasks. To correctly relay the progress from the look-ahead plan to the master schedule, it is essential to understand to what extent does finishing each task on the look-ahead plan contribute towards finishing an activity on the master schedule.
- Creating a user-friendly interface and a document management infrastructure that enables the upload of documents and the output of results.
- Enabling a feedback loop that allows the user to confirm the relayed progress as well as correct it when necessary.

The presented use case contains multiple technical challenges. In this manuscript, we present a model that attempts solving the first step. In other words, we only focus on the first piece of the workflow which is related to establishing the links between master schedule activities and look-ahead plan tasks, and we highlight the remaining steps as future research opportunities to be tackled. The following sections detail our approach for automating the process of mapping look-ahead plan tasks to master schedule activities in detail. We present our model, present our results, and discuss the contributions and limitations of our approach.

3.3. Distance-based matching baseline model

Before designing a modern supervised machine learning model, we explored the application of simple baseline methods such as distance-based models to map activity and task representations to one another. To do so, task descriptions are first pre-processed for natural language processing needs by making them lowercased, tokenized, and lemmatized. Similarly, all information related to a schedule activity (concatenated WBS tags and activity description) is pre-processed similar to tasks, and then they are concatenated together in one phrase (see Table 1 in Section 4.1). Second, we describe activities and tasks using FastText embeddings [35] trained on data from 32 construction schedules. As described in Amer and Golparvar-Fard [19], FastText embeddings are used to vectorize both task and activity descriptions and project them into a numerical space. Each activity and task is represented as the mean of the embeddings of all the words that constitutes it as shown in Fig. 4. Third, Cosine similarity in Eq. (1) is used to identify the closest activity description to each task description. If a task has the same maximum similarity to two activities, both activities are assumed to be correct matches. Cosine similarity values range between −1 and 1 where −1 indicates that the task and activity are too distant (opposite), while 1 indicates that they are identical.

$$\text{Similarity}(w_i, a_j) = \frac{w_i \cdot a_j}{\|w_i\|\|a_j\|} = \frac{\sum_{l=1}^{d_k} w_{il} a_{jl}}{\sqrt{\sum_{l=1}^{d_k} w_{il}^2}\sqrt{\sum_{l=1}^{d_k} a_{jl}^2}} \quad (1)$$

where $d_k = 320$ and it represents the size of vectors w_i and a_j. The value of the embedding size d_k is predefined and validated in previous works such as [19].

3.4. Location extraction and matching

Locations play a significant role in construction planing, and they are identified as one of the constraining flows of construction [36]. When creating a master schedule, location information is often embedded in the WBS as well as in the activity description. Similarly, when building look-ahead plans, the work is split by location to manage production and crew movement. In this section, we leverage the location information available through both master schedule activity and look-ahead planning task descriptions to improve the alignment between the two. To do so, we first extract the locations of both task and activity using the Part-of-Activity tagging method presented in Amer and Golparvar-Fard

Table 1
Examples from the data showing matches between master schedule activities and look-ahead planning tasks where an activity can be associated with multiple tasks.

Master Schedule		Look-Ahead Plan
WBS 1 > WBS 2	**Activity Description**	**Task Description**
Construction Tower > Floor Fit-outs	Tower Finishes 7	LEVEL 07 - TAPE AND FLOAT
Construction Tower > Floor Fit-outs	Tower Finishes 7	LEVEL 07 - TUBS/ SHOWERS
Construction Tower > Floor Fit-outs	Tower Finishes 7	LEVEL 07 - WALL COVERINGS
Construction Tower > Floor Fit-outs	Tower Finishes 7	LEVEL 07 - WALL/ FLOOR TILE
Construction > Building Envelope	EIFS	LEVEL 08 - DRY-IN
Construction > Building Envelope	EIFS	LEVEL 08 - EIFS- FINISH COAT
Construction > Building Envelope	EIFS	LEVEL 08 - EIFS- FOAM
Construction > Tower MEP	MEP Risers	LEVEL 08 - FIRE PROTECTION BRANCHES
Construction > Tower MEP	MEP Risers	LEVEL 08 - FIRE PROTECTION MAINS
Construction > Tower MEP	MEP Risers	LEVEL 08 - FIRE PROTECTION STANDPIPE

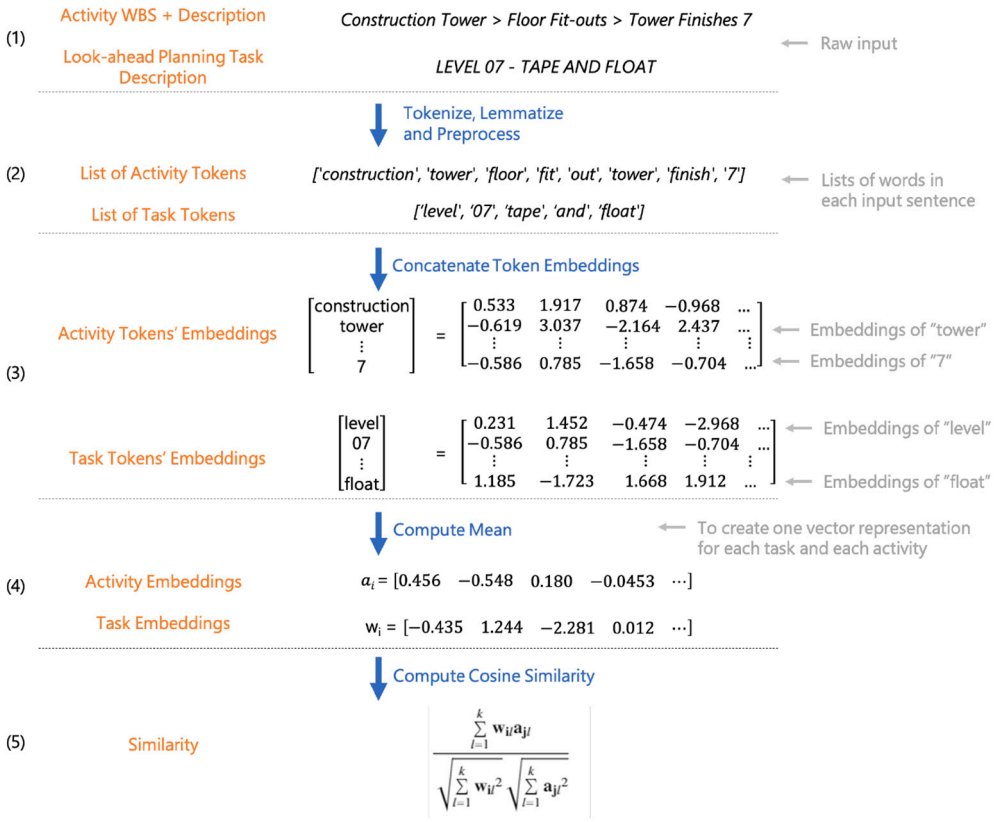

Fig. 4. Distance-based similarity computations between activity and task descriptions.

[24]. Second, we normalize location information by transforming (a) ("*floor*", "*fl*", "*l*", "*lvl*") to "*level*" and ("*first*", "*second*", etc..) to ("*1*", "*2*", etc.) respectively. Third, we perform fuzzy matching using Levenshtein distance (Eq. (2)):

$$\mathrm{lev}(w_{\mathrm{loc}}, a_{\mathrm{loc}}) = \begin{cases} |w_{\mathrm{loc}}| & \text{if } |a_{\mathrm{loc}}| = 0 \\ |a_{\mathrm{loc}}| & \text{if } |w_{\mathrm{loc}}| = 0 \\ \mathrm{lev}(\mathrm{tail}(w_{\mathrm{loc}}), \mathrm{tail}(a_{\mathrm{loc}})) & \text{if } w_{\mathrm{loc}}[0] = a_{\mathrm{loc}}[0] \\ 1 + \min \begin{cases} \mathrm{lev}(\mathrm{tail}(w_{\mathrm{loc}}), a_{\mathrm{loc}}) \\ \mathrm{lev}(w_{\mathrm{loc}}, \mathrm{tail}(a_{\mathrm{loc}})) \\ \mathrm{lev}(\mathrm{tail}(w_{\mathrm{loc}}), \mathrm{tail}(a_{\mathrm{loc}})) \end{cases} & \text{otherwise} \end{cases} \tag{2}$$

where w_{loc} and a_{loc} are the task and activity locations respectively, and *tail(expression)* is the *expression* without its first character. For example, *tail("level 02")* is "*evel 02*". Last, we return the list of matching-by-location candidates to be processed by the distance-based matching model.

3.5. Supervised neural model

In this section, we introduce our new supervised model which (1) predicts pairwise relationships between look-ahead planning tasks and master schedule activity descriptions, and (2) generates a list of look-ahead planning tasks given an input master schedule activity. To achieve these goals, we employ a novel Decoder Transformer architecture [37], because transformers have shown superior performance on ranking problems when compared to other approaches [12]. Instead of training a classifier that takes task and activity descriptions as inputs and

predicts their relationship from scratch, our approach fine-tunes the small version of the OpenAI's GPT-2 language model [14], which is pretrained over 40GB of Internet text. Our model can be defined as a mapping function "Matcher" (*M*) that maps all tuples of ⟨ *task, activity* ⟩ into the binary space {0,1}:

$$M : \mathbb{W} \times \mathbb{A} \rightarrow \{0, 1\} \tag{3}$$

$$l_{ij} = M(w_i, a_j) \text{ s.t. } l_{ij} \in \{0, 1\} \tag{4}$$

where $\mathbb{W}$ is the space of all possible tasks, $\mathbb{A}$ is the space of all possible activities, and {0,1} refer to {*No relationship, Relationship*} respectively. In other words, the model performs a binary classification, and the possible classes are either "Relationship" (indicating that the master activity is a correct match for the look-ahead plan task) or "No Relationship" (indicating that the master activity is not a correct match for the look-ahead plan task). Since both activities and tasks are expressed using text strings and English expressions, we denote $\mathbb{V}$ to be the set of all English expressions (or vocabulary) used to describe an activity or task. Function *M* can then be reformulated as the mapping function between $\mathbb{V}$ and {0,1} such as:

$$M : \mathbb{V}^2 \rightarrow \{0, 1\} \tag{5}$$

Before diving into the details of the supervised model, we first provide a high-level overview of Language Models (LMs) and pre-trained language representations, as follows:

3.5.1. Language modeling

Language models – specifically forward language models – calculate the probability of a sentence (represented as a list of tokens: $t_1, t_2, \ldots, t_{k_N}$) by computing the probability of the next token (t_k) given all previously seen tokens ($t_1, t_2, \ldots, t_{k-1}$) as shown in Eq. (6) [38]. For example, given an activity description "L27 Reshoring Pour Deck & Columns" as an input sentence, a language model calculates the probability of "Reshoring" given "L27", "Pour" given "L27 Reshoring", "Deck" given "L27 Reshoring Pour", etc.

$$P(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} P(t_k | t_1, t_2, \ldots, t_{k-1}) \tag{6}$$

Language models play an essential role in traditional NLP tasks such as text summarization, machine translation, and speech recognition. These models learn information encoded in the training data in a compact form and they can better represent input data for downstream tasks compared to traditional NLP pipelines [39]. Earlier *N*-gram language models relied on probability distributions conditioned on *N* previous tokens in the sentence. For example, Unigram language models use only one token as context to predict the next token. The probability distribution of an N-gram language model is saved in static tables which are used later for prediction [40].

The second generation of language models is neural. These models use context-independent word embeddings –such as Word2Vec [41] – to represent input tokens, and then a variance of a Recurrent Neural Network (RNN) such as a Gated Neural Network (GNN) or a Long Short Term Memory (LSTM), to condition the prediction on the current context [40]. These recurrence-based models are often enhanced with attention mechanisms on their learned internal representations to be used for downstream tasks such as neural machine translation [42]. These attention mechanisms allow the model to pay specific attention to specific tokens of an input sentence when predicting different tokens in the translated sentence. However, the sequential nature of recurrent models prevents parallelization within training examples and creates training limitations especially when parsing longer sentences.

The Transformer architecture [43] removes the reliance on recurrence and uses only attention mechanisms to perform machine translation tasks using less training time while achieving better performance. This performance introduced the Transformer as the basic building block for state-of-the-art language models and other NLP tasks. Transformers enabled language models to be trained on vast amounts of text data [44,45]. Without supervision, these language models achieved state-of-the-art results on many downstream NLP tasks such as question answering and named entity recognition with little or no fine-tuning. In this paper, we leverage the capabilities of a Transformer decoder architecture known as the GPT-2 model – small version – developed by OpenAI [14]. We fine-tune the model to (1) predict whether pairwise links should be established between any given input master schedule activity and look-ahead planning task, and (2) generate a list of look-ahead planning tasks given an input master schedule activity.

3.5.2. Multi-head attention and transformer decoder blocks

GPT-2 is composed of 12 layers of Transformer Decoder blocks [37]. These blocks enable GPT-2 to train on long sequences by removing the encoder module from the original transformer architecture which reduces the total number of model parameters by half. This approach models each input example as one sequence that includes both the training example and its label. As shown in Table 1 (see Section 4.1), each training example is composed of two levels of WBS description in the master schedule, the activity description, and the look-ahead planning task description. To train the Transformer, each example is fed to the model in the form of a sentence as shown in Fig. 5. As mentioned earlier, the master activity information is composed of two WBS descriptions and an activity description. The special token "<SEP>" is used to separate these entities in training and testing examples. Similarly, the special token "<EOS>" is used to declare the end of both activity and task descriptions.

As shown in Fig. 6, each Decoder block applies a multi-headed self-attention mechanism followed by a fully connected layer to create a distribution over the target vocabulary:

$$h^0 = \text{LayerNorm}(\text{XW}_e + W_p) \tag{7}$$

$$h^{(i)} = \text{block}(h^{(i-1)}) \forall i \in [1, n = 12] \tag{8}$$

where *X* is the input sentence, W_e holds the weights of the word embeddings, W_p contains the input positional embeddings which provide information about the position of each token in the input training example, *LayerNorm* is a normalization function to regulate the gradients during the training process, h^0 is the output embedding that is fed to GPT-2, $h^{(i)}$ is the output of decoder block *i*, and *block* is the mapping function performed by a decoder block and is described by the following:

$$h^{(i)} = \text{LayerNorm}\left(\text{Attention}(Q, K, V) + \text{Residual}(h^{(i-1)})\right) W_D \tag{9}$$

where W_D holds the weights of the fully connected layer of the decoder block, *Residual* is a bypass that enables downstream decoder blocks to digest the original model input, and *Attention* is the transformation applied by the masked self-attention block (see Fig. 7 for an illustration of the performed computations) and is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{\text{QK}^T}{\sqrt{d_k}}\right) V \tag{10}$$

where d_k is the size of the input vector embeddings, *Q, K, V* refer to "*Queries*", "*Keys*", and "*Values*" which are abstractions that enable the model to perform the attention computations. The "*Queries*" refer to the input vector to the decoder block after being transformed using a learnable fully connected layer having the weight matrix W_Q:

$$Q = h^{(i-1)} W_Q \tag{11}$$

Similarly, "*Keys*" refer to the matrix containing a transformed version of the input vector to the decoder block $h^{(i-1)}$ repeated *n* times where *n* is the size of $h^{(i-1)}$. To create "*Keys*", $h^{(i-1)}$ is transformed using a fully connected layer having the weight matrix W_K such that:

Fig. 5. Training example composed of master activity details, look-ahead plan task details, and a label that indicates the relationship between the two. The master activity is described using two levels of WBS and the activity description. The special token <SEP> is used to separate the different components of the activity description, and the special token <EOS> is used to indicate the end of both the activity and the task.

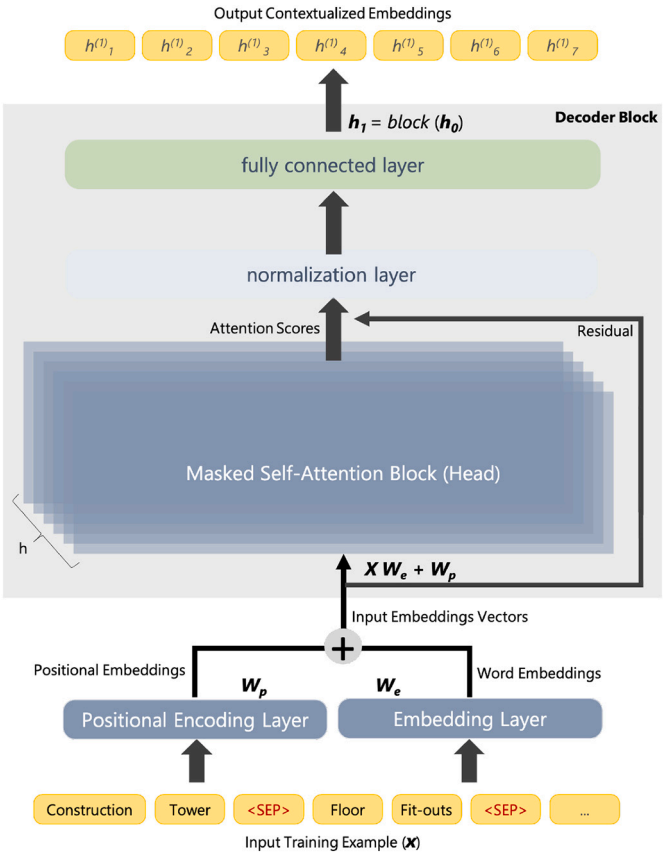

Fig. 6. The input, output, and architecture of the first decoder block in the GPT-2 model.

$$K = h^{(i-1)} W_K \tag{12}$$

This formulation of "*Queries*" and "*Keys*" enables the computation of self attention scores by simply computing QK^T as shown in Eq. (10). Lastly and similar to "*Queries*", "*Values*" is a transformed version of $h^{(i-1)}$ (Eq. (13) where W_V is the weight of the fully connected layer), and it enables the application of the computed scores (QK^T) to be applied to the input vector $h^{(i-1)}$ as described by Eq. (10).

$$V = h^{(i-1)} W_V \tag{13}$$

As described by Fig. 7, within every block and for every embedding at position j in the block's input, attentions are learned and computed over all embeddings at positions [0,...,j]. The masks shown in Fig. 7 demonstrate how the model is unidirectional. The output embeddings of block i are then fed into block $i+1$ as shown by Eq. (8).

To enable our model to perform both pairwise predictions and task description generation, at the end of the computations of the last decoder block, we channel the final outcome $h^{(12)}$ into (1) a language modeling head (LM_{head}) and (2) a classification head (C_{head}) (Fig. 8). The LM_{head} consists of a fully connected layer followed by a *Softmax* layer such that:

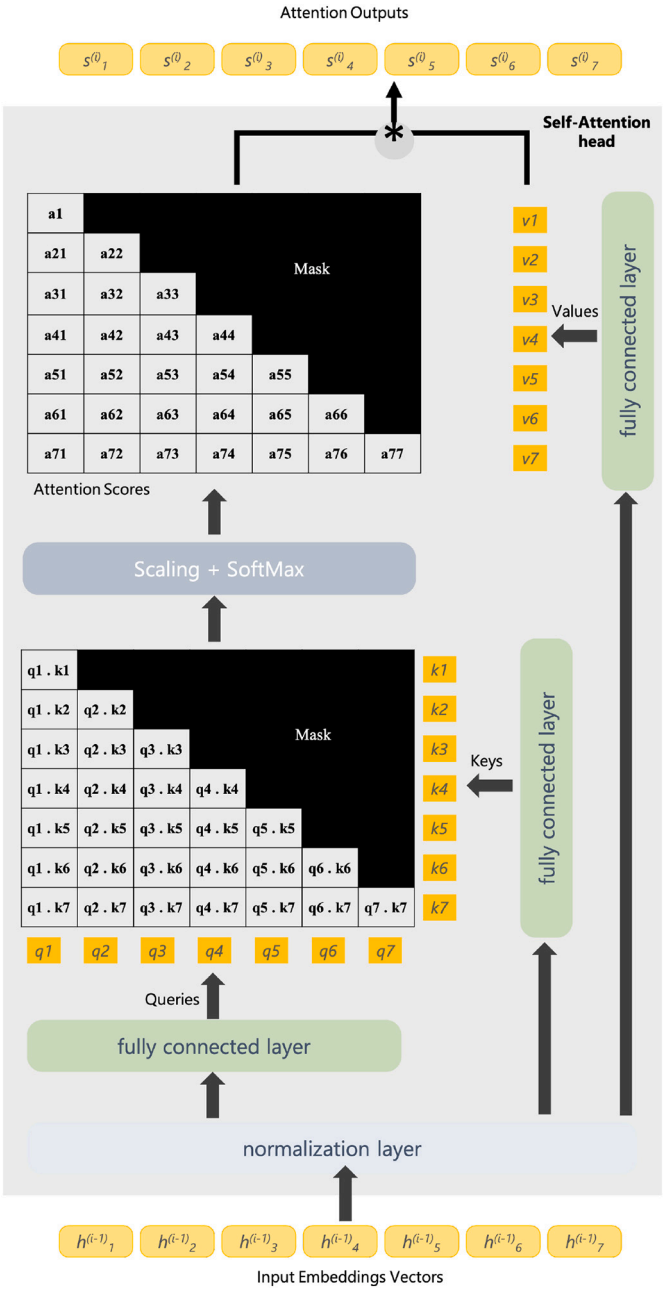

Fig. 7. Computations performed by a Decoder Transformer self-attention head.

$$\mathrm{LM}_{\mathrm{head}} : Z_{\mathrm{lm}} = \mathrm{Softmax}(h_{12} W_{\mathrm{LM}}) \tag{14}$$

where W_{LM} are the weights of the LM_{head}'s fully connected layer and Z_{lm} is the probability distribution over every word in the vocabulary (50,260 words in total) at every position i in the input sentence given all previous positions $[0, \ldots, i-1]$. This is used to teach the model to generate look-

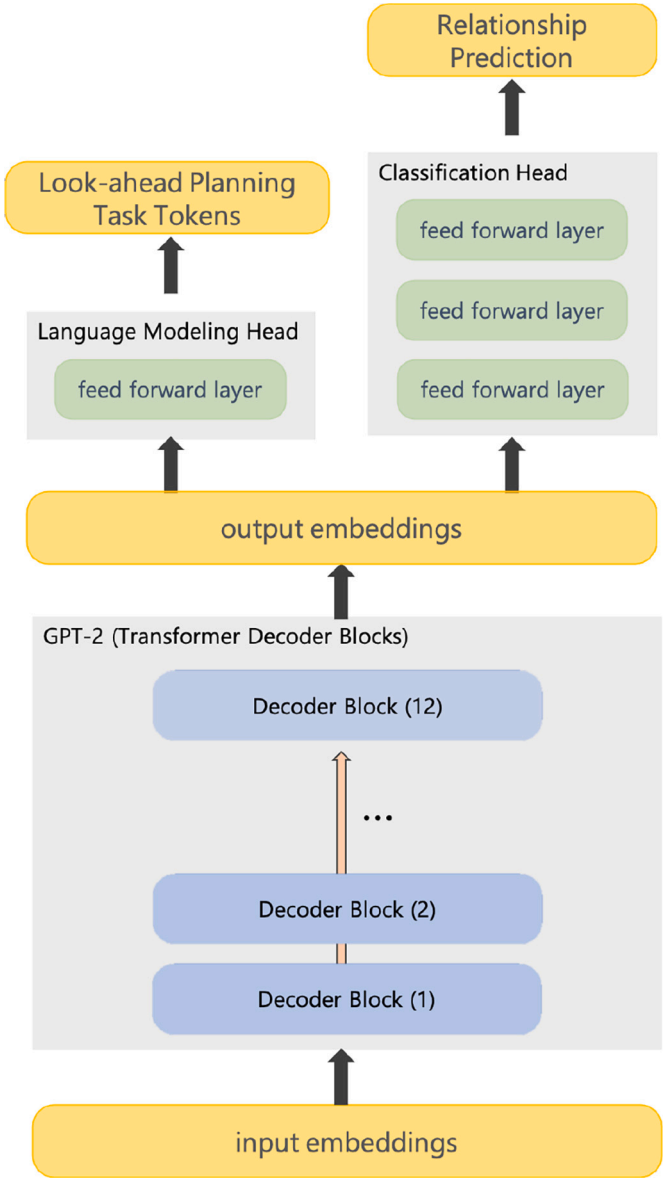

Fig. 8. Supervised Transformer model architecture. The GPT-2 transforms the input information of look-ahead planning tasks and master activities into the embedding space, the language modeling head learns to generate task descriptions given a master schedule activity, and the classification head learns to predict the relationship status between the task and the activity.

ahead planning task descriptions given the description of a master schedule activity. As discussed later in Section 4.1, the training data have both positive examples where the look-ahead planning task is related to the master activity and negative examples where the task and activity are independent. To train the language modeling head, we only update its parameters when the model parses positive examples so that it learns dependencies only from examples where the task is linked to the activity.

On the other hand, the C_{head} is trained on both positive and negative examples and it performs a binary classification task as described earlier by Eq. (3). It is composed of three fully connected layers validated empirically, and it employs *LeakyRelu* activation functions followed by a *Softmax* layer as follows:

$$Z_{c_1} = \text{LeakyRelu}(h_{12} W_{c_1}) \tag{15}$$

$$Z_{c_2} = \text{LeakyRelu}(Z_{c_1} W_{c_2}) \tag{16}$$

$$Z_{c_3} = \text{LeakyRelu}(Z_{c_2} W_{c_3}) \tag{17}$$

$$Z_c = \text{Softmax}(Z_{c_3}) \tag{18}$$

where W_{c_1}, W_{c_2} and W_{c_3} are the weights of the fully connected layers and LeakyRelu for an input x is a defined as:

$$\text{LeakyRelu}(x) = \max(0, x) - 0.01 \times \min(0, x) \tag{19}$$

The classification performed by the model is a pairwise judgment of the likelihood that a look-ahead planning task w_i matches a master activity a_j. However, the ultimate goal is to identify the most relevant master activity for any input task. In other words, for each task, the model is used to perform a pairwise judgment of the likelihood of matching every available master activity. The pairwise judgements are then used for ranking different activities. The scores of the classification head (Z_c in Eq. (18)) for the activities predicted by the model to be positive matches for the input task are used to rank the relevance of the activities with respect to the input task. Those scores represent the probabilities of being a correct match as calculated by the Softmax function). A higher score indicates a higher confidence by the model that the positive prediction is correct. This is a typical use of pairwise prediction models for ranking [12]. This approach for ranking is useful to filter out false positive predictions and to insure the returned results are those judged by the model to be the fittest matches.

3.6. Multi-stage architectures

As opposed to using different models in isolation, the distance-based model, the location-based matching, and the supervised transformer model were used together in hierarchical multi-stage architectures to improve matching performance and to validate design choices. For instance, we test the following combinations, and we select the model combination that achieves the best Precision, Recall, and f1 Score:

- Candidates generation using location-based matching followed by distance-based matching
- Candidates generation using location-based matching followed by transformer-based matching
- Candidates generation using distance-based matching followed by transformer-based matching
- Candidates generation using both location-based and distance-based matching followed by transformer-based matching

The next sections discuss the data used for validating the model, the quantitative and qualitative results, and the contributions and limitations of the presented method.

4. Experimental setup, results and discussion

4.1. Data

The experimental data is composed of master schedules of four different commercial building projects and 30 look-ahead plans that cover partial periods of those projects. Overall, 2131 logic links were manually established between 460 master schedule activities and 1905 look-ahead planning tasks by three expert annotators (project planners) so that the ground truth can be properly quality controlled. The relationship between look-ahead planning tasks and master schedule activities is many-to-one where only one master schedule activity can be linked to a look-ahead planning task while many look-ahead planning tasks can be linked to the same master activity. Table 1 shows examples of the activity-vs.-task alignments. It can be seen that among these pairs of activities and tasks, the WBS, level of detail, and expressions are different.

The initial data was only composed of activity-vs.-task links that

should exist (positive examples). However, to enable training the supervised model, the data was augmented to include negative examples where the look-ahead planning task and master schedule activity are irrelevant and should not be linked together. To do so, we randomly sample one task and one activity and flag their links as negative if it was not seen in the list of positive links within each of the four available projects. This is only possible because the annotators insured the list of positive links is exhaustive; i.e., all links that should be established between look-ahead planning tasks and master schedule activities are identified and approved through consensus among three expert annotators. Overall, the number of generated negative examples was 4262, i.e., two negative examples for every positive example. This ratio was set and validated empirically based on trial and error.

The data is split into five validation folds. Each fold contains training and testing data from all four projects. At test time, for each look-ahead planning task, the master schedule activities of the same project are ranked based on relevance according to the model used for testing. As mentioned in Section 3, we test the performance of our method on fully automated matching by returning the single correct master schedule activity match for each look-ahead planning task. We also test the method's fitness to be part of a human-in-the-loop approach where the model returns the top five matches and leaves the final decision to the project planner to pick the correct match.

4.2. Supervised GPT-2 model pairwise prediction results (pre-ranking)

First, we share results on the performance of the supervised GPT-2 transformer model on the pairwise classification task. As shown in Table 2, the model can correctly predict a positive match with Precision, Recall, and f1-Score of 82.1%, 78.54%, and 80.1% respectively. Similarly, the model can predict a correct negative relationship (accurate prediction of a "No Relationship") with Precision, Recall, and f1-Score of 89.4%, 91.1%, and 90.2% respectively.

Table 3 shows qualitative examples of the pairwise predictions made by the model. Examples 1 and 2 show accurate predictions. Examples 3 and 4 show false negative predictions where the ground truth is positive but the prediction is negative. And, Examples 5 and 6 show false positive predictions where the ground truth is negative but the prediction is positive.

4.3. Location-based matching results

Table 4 shows examples of the performance of ALOR tagging [24] in terms of identifying location related terms in both look-ahead planning tasks and master schedule activities. The table also highlights the performance of Levenshtein distance-based location matching model in identifying potential matching activities that have the same location as the look-ahead planning task under consideration.

4.4. Ablation study and ranking results

The goal of the ablation study is to explore whether the process of linking look-ahead planning tasks to master schedule activities can be expedited to help site engineers and superintendents review and then align their short-term and long-term plans for production planning and reporting purposes. To that end, we test two different approaches: (1) The first approach attempts the full automation of the task to activity matching process by identifying the most relevant master activity for any given look-ahead planning task. (2) The second approach examines semi-automating the process following a human-in-the-loop approach where the model suggests to the engineer a list of top-five activities that are most relevant to a look-ahead planning task. For each approach, we deploy and test six different model designs which are all validated following a five-fold validation schema as discussed in Section 4.1. A discussion on the reasons for considering a semi-automated approach is further discussed in Section 5.

Table 2
Pairwise predictions Precision, Recall, and f1 Scores of the supervised GPT-2 Transformer model across five validation folds.

	Precision		Recall		f1 Score	
	μ	σ	μ	σ	μ	σ
Relationship	0.821	0.025	0.785	0.051	0.801	0.018
No Relationship	0.894	0.018	0.911	0.021	0.902	0.004

Table 5 shows the results of the first approach, i.e., the mean and standard deviations of Precision, Recall and f1 Score for returning the exact correct master activity match for a given look-ahead planning task, across all validation folds. The best performing design was the distance-based model followed by the Transformer model achieving a Precision, Recall, and f1 Score of 51.1%, 40.3%, and 44.9% respectively. On the other hand, Table 6 shows the results of the second approach where Precision, Recall, and f1 Score are measured based on whether the correct match appears in the returned list or not. Similar to the fully automated approach, the best performing design was the distance-based model followed by the Transformer model which achieved a Precision, Recall, and f1 Score of 76.5%, 64.3%, and 69.8% respectively. The dataset we used for training and validation contained a master schedule activity match for each look-ahead planning task. The validation on our model's ability to identify tasks that do not have a corresponding master activity is therefore limited to the results of the pairwise predictions shown in Table 2 where the f1 Score on predicting a negative relationship is 90.2%.

As shown in Tables 5 and 6, some of the tested model designs use multistage ranking architectures such as "Distance + Transformer" where the Distance-based model is used to generate candidate matches for the Transformer model which then fine-tunes the results. To identify the ideal number of candidates to be generated by the distance-based model, we tested different configurations on one of the models (the model tested on validation fold #1) and we show the results in Figs. 9 and 10 for the first and second approaches respectively. The best results were achieved by setting the number of candidates to ten for the first approach and 25 for the second approach. This number was then fixed for all the experiments across all validation folds.

In addition to reporting the performance of different models on the matching problem, we also investigated the amount of time required by different models to match a single look-ahead planning task. This is essential to understand the usability of different models and their ability to scale to actual industry requirements. Table 7 shows the time in milliseconds required to match a single look-ahead planning task per master activity. This time is the average recorded across all four projects and all five validation folds. The best performing model – Distance + Transformer – required 0.108 ms per look-ahead planning task per activity. In other words, for a look-ahead plan of 50 tasks and a master schedule of 2000 activities, the required overall time is 10.8 s.

4.4.1. Generation of tasks from activities

As mentioned in Section 3, our supervised model has a generative language modeling head with the ability to generate look-ahead planning tasks using a master schedule activity as input. To do so, our model ingests the description and WBS tag of a master schedule activity and returns a list of look-ahead planning tasks that has the highest probability of matching that activity. As defined in Eq. (14), task descriptions are generated at an individual word level; i.e., the model generates the description of every look-ahead planning task word-by-word as opposed to selecting a full look-ahead planning task description from a predefined list of tasks. Given the sequence of words in the master activity WBS and description, the model maintains the smallest between (a) the list of top k-words [46] where k is empirically set to 25 or (2) the list of words whose sum of probabilities is equal to a p-value [47] of 0.9. Urging the probabilities of the top list of words to be above 0.9 causes the model to generate less than five sentences in the cases where that

Table 3
Qualitative examples of the predictions made by the supervised GPT-2 model.

Example	Model Input	Model Prediction	Ground Truth
(1)	Interior Build out _ Apartment Finishes <SEP> Level 24 <SEP> MEPFP Overhead Rough Level 24 <EOS> MEPFP_Level 24 OH Rough-In <EOS>	1 (Relationship)	1 (Relationship)
(2)	Construction <SEP> Site finishes <SEP> Pool Construction <EOS> L11 Deck_Install Tables <EOS>	0 (No Relationship)	0 (No Relationship)
(3)	MEP <SEP> MEPFP RISERS <SEP> MEPFP Risers Level 1 <EOS> Fire Protection _level 1 stair 2 & West FDC riser <EOS>	0 (No Relationship)	1 (Relationship)
(4)	<SEP> Parking Garage Structure <SEP> Masonry (All Levels) <EOS> Garage Side>Topping Slabs/CMU Work_Pour B1 North <EOS>	0 (No Relationship)	1 (Relationship)
(5)	Interior Build out _ Apartment Finishes <SEP> Level 30 <SEP> MEPFP Overhead Rough Level 30 <EOS> Mechanical_Level 21 OH Rough-In <EOS>	1 (Relationship)	0 (No Relationship)
(6)	<SEP> Exterior/Sitework <SEP> Streetscape NorthWest <EOS> Interior Framing _Level 8 Drywall <EOS>	1 (Relationship)	0 (No Relationship)

Table 4
Examples of location extraction and matching between look-ahead planning tasks and master schedule activities.

Look-ahead Planning Task	Master Activity
level 32 vertical work	superstructure I 8 to I 36 concrete level 32
	superstructure I 8 to I 36 I 32 reshoring pour deck and column
	superstructure I 8 to I 36 I 32 slab mepfp sleeve insert rough in
	masonry - masonry level 2
	masonry - masonry level 3
1 bldg b structure rebar wall level 03 04	form rebar and place wall column bldg b level 03 04
	form rebar and place deck bldg b level 04
	drywall wall and ceiling bldg b level 03
	form rebar and place wall column bldg b level 13 14
	form rebar and place wall column bldg b level 04 05
level 05 hvac ductwork	construction tower <SEP> floor fit out <SEP> tower finish 5
	construction tower <SEP> floor fit out <SEP> tower roughing 5 <EOS>
	construction tower <SEP> floor fit out <SEP> tower roughing 5 <EOS>
	construction tower <SEP> superstructure shell construction <SEP> column 4 pt slab 5 <EOS>
	construction tower <SEP> superstructure shell construction <SEP> column 5 pt slab 6 <EOS>

Table 5
Precision, Recall, and f1 Score of all six tested model configurations **on identifying the most relevant master activity to an input look-ahead planning task**. (a) The "Distance Only" model uses Cosine similarity and pre-trained FastText embeddings, (b) the "Location + Distance" uses Location matching to generate candidates for the distance-based matching model. (c) The "Transformer Only" model uses the supervised transformer model to perform pairwise predictions on the relationship of the input look-ahead planning task and every available master activity and then ranks the results based on the probability of a positive relationship. (d) The last three models "Location", "Distance", and "Location + Distance" matching to generate candidates for the Transformer model which then performs the final ranking.

	Precision		Recall		f1 Score	
	μ	σ	μ	σ	μ	σ
Distance Only	0.328	0.009	0.328	0.009	0.328	0.009
Location + Distance	0.320	0.015	0.320	0.015	0.320	0.015
Transformer Only	0.282	0.032	0.300	0.021	0.289	0.015
Location + Transformer	0.387	0.019	0.318	0.042	0.348	0.031
Distance + Transformer	**0.511**	0.022	**0.403**	0.027	**0.449**	0.013
Location + Distance + Transformer	0.516	0.034	0.384	0.041	0.439	0.034

The values in Bold are those of the best performing model.

requirement is not satisfied for a large pool of words. The probability mass is then redistributed over the list of top words, and a second word is chosen accordingly. Additionally, since multiple look-ahead planning tasks can be associated with one master schedule activity, the model performs multiple simultaneous searches for the next word in parallel to ensure the generation of multiple look-ahead planning tasks. In our experiment, we empirically set the number of simultaneous searches to 50. In each parallel search, the chosen word is added to the input used to generate the following word until the special token "<EOS>" is encountered or a maximum task length of 50 is reached. Since task descriptions are often short, this max task length works as an upper limit and is almost never encountered. Among the 50 generated task descriptions, the model picks the top five that achieve the highest prediction scores. The values of the hyper-parameters chosen for the generation task are all based on empirical testing, and the degree of their influence on the quality of the results can be further investigated in

Table 6
Precision, Recall, and f1 Score of all six tested model configurations **on identifying a list of five master activities that contain the most relevant master activity to an input look-ahead planning task**. (a) The "Distance Only" model uses Cosine similarity and pre-trained FastText embeddings, (b) the "Location + Distance" uses Location matching to generate candidates for the distance-based matching model. (c) The "Transformer Only" model uses the supervised transformer model to perform pairwise predictions on the relationship of the input look-ahead planning task and every available master activity and then ranks the results based on the probability of a positive relationship. (d) The last three models "Location", "Distance", and "Location + Distance" matching to generate candidates for the Transformer model which then performs the final ranking.

	Precision		Recall		f1 Score	
	μ	σ	μ	σ	μ	σ
Distance Only	0.650	0.021	0.650	0.021	0.650	0.021
Location + Distance	0.616	0.019	0.616	0.019	0.616	0.019
Transformer Only	0.634	0.028	0.569	0.029	0.599	0.023
Location + Transformer	0.652	0.025	0.494	0.037	0.561	0.027
Distance + Transformer	**0.765**	0.023	**0.643**	0.043	**0.698**	0.028
Location + Distance + Transformer	0.712	0.026	0.568	0.050	0.631	0.038

The values in Bold are those of the best performing model.

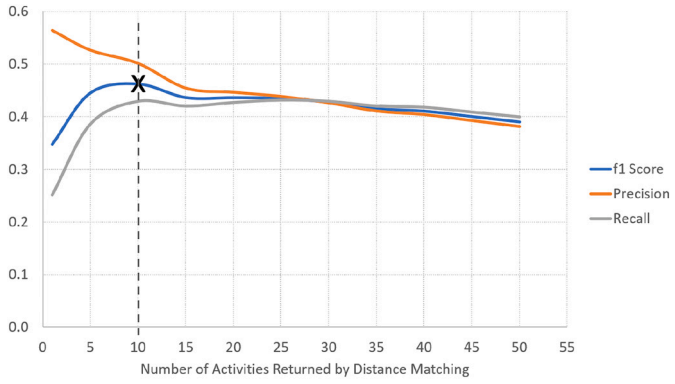

Fig. 9. Precision, Recall, and f1 Score of the "Distance + Transformer" model on identifying the most relevant master activity to an input look-ahead planning task based on the number of candidates generated by the distance-based matching model.

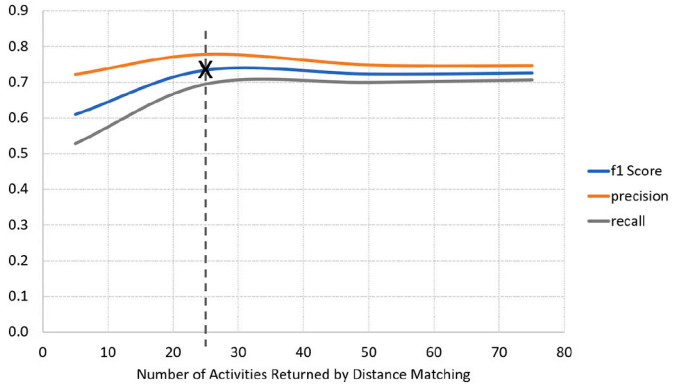

Fig. 10. Precision, Recall, and f1 Score of the "Distance + Transformer" model on returning a list of five master activities which contain the most relevant activity to an input look-ahead planning task based on the number of candidates generated by the distance-based matching model.

future research efforts. Table 10 shows test examples of tasks generated by the model given activity description prompts. For instance, the first example shows a list of tasks generated based on the activity "*finishes level 4*". In The next section, we discuss the results and explain our thought process and design considerations.

5. Discussion

Although the embeddings used to perform distance-based matching are trained on construction scheduling specific data, the distance model achieved an f1 Score of 32.8%. Table 8 shows qualitative ground truth examples (positive examples) from the data where the similarity between tasks and activities is minimal. These examples highlight the difference in granularity between look-ahead planning task descriptions and activity descriptions where tasks are more fine-grained while activities are more general and encompassing. It also highlights the differences in the language used by different project stakeholders who create the master schedule (project planners) and the look-ahead plans (project superintendents). Table 9 (Examples 1, 2, and 3) show cases where the similarity in describing activity and tasks yielded correct model predictions. On the other hand, Table 9 (Examples 4, 5, and 6) show cases of inaccurate predictions made by the distance-based model, i.e., examples where the similarity is the highest but the matching is inaccurate. For instance, in Example 4, the word "*deck*" in "*steel and deck*" caused the model to inaccurately predict "*Steel and Deck*" as the master activity match to "*L26 deck install tables*" although the WBS information of "*steel and deck*" shows that this activity is not being undertaken in "*L26*" (26th floor) but in "*Bridge*". Similarly, in example 5, the ground truth master schedule activity "*elevator > lagging*" only has one word in common with the look-ahead planning task description, while the predicted activity is more similar. Lastly, Example 6 shows a case where the embedding model failed to capture the similarity between "*MEPFP*" and "*fire protection*" and instead assumed "*masonry*" and "*stair*" to be more similar causing the predicted master activity to be inaccurate. This example also highlights the sensitivity of the distance-based model to the length of the task and activity descriptions where extra words can cause the similarity between tasks and activities to decrease even if they both contain other similar expressions.

Table 7
Average time in milliseconds required to retrieve the master activity match for a single look-ahead planning task.

	Time per Task per Activity (ms)
Distance Only	0.130
Location + Distance	0.058
Transformer Only	0.891
Location + Transformer	0.245
Distance + Transformer	**0.182**
Location + Distance + Transformer	0.108

By checking the results of the "Location + Distance" model in Table 5, we can see that adding a filtering by location step before distance-based matching performed worse compared to the distance-based matching alone with an f1 Score of 32%. Since location-based matching is performed using Levenshtein distance measurements, it has a similar behavior to the FastText embeddings used by the distance-based model where both models consider sub-word representations. The differences are that (a) location-based matching is performed only on

Table 8
Examples showing ground truth alignments between master schedule activities and look-ahead planning tasks where the activities and tasks have little similarity between their descriptions.

Master Schedule		Look-ahead Plan
WBS 1 > WBS 2	**Activity Description**	**Task Description**
Superstructure > L8 to L36	CONCRETE LEVEL 26	L26 Deck Vertical Work
Superstructure > L8 to L36	CONCRETE LEVEL 26	L26 Deck Install Tables
Bldg B > Level 14	Form, Rebar & Place Deck	Bldg B: Structure Post-Tension Cables
ELEVATOR	LAGGING	Elevator Jump #3 Rails to level 33
Superstructure > Roof	CONCRETE ROOF LEVEL 37	Level 37 Embeds

The values in Bold are those of the best performing model.

Table 9
Qualitative examples of the predictions made by the distance-based model.

Example	Look-ahead Planning Task Description	Master Activity (Ground Truth)		Master Activity (Distance-based Model Prediction)	
		WBS 1 > WBS 2	**Description**	**WBS 1 > WBS 2**	**Description**
(1)	Masons Level 1	MASONRY	Masonry Level 1	MASONRY	Masonry Level 1
(2)	L8 Deck South Half Top embeds	Superstructure > L8 to L36	CONCRETE L 8 Deck	Superstructure > L8 to L36	CONCRETE L 8 Deck
(3)	Bldg B: Structure Post-Tension Cables	Bldg B > Level 11	Form, Rebar & Place Deck	Bldg B > Level 11	Form, Rebar & Place Deck
(4)	L26 Deck Install Tables	Superstructure > L8 to L36	CONCRETE LEVEL 26	Bridge Set	Steel and Deck
(5)	Elevator Jump #3 Rails to level 33	ELEVATOR	LAGGING	Superstructure > L8 to L36	CONCRETE LEVEL 33
(6)	Fire Protection level 1 stair 2 & West FDC riser	MEP > MEPFP RISERS	MEPFP Risers Level 1	MASONRY	Masonry Level 1

location terms while distance-based matching is performed on the full task and activity descriptions, and (b) FastText embeddings learn the "distributional similarity" between words that do not share common characters – i.e., the similarity based on where words and sub-word representations appear in training data – while Levenshtein distance measurements only compare word characters. As a result, if the correct master activity match has a completely different location term than the look-ahead planning task, it will be filtered out by the location-based matching step before the distance-based model can assess its similarity to the look-ahead planning task even if all other terms that describe the activity are similar to those that describe the task. On the other hand, using location-based matching before ranking using the Transformer model improved the results from 28.9% to 34.8% by filtering out wrong candidates. The best results where achieved by generating a list of candidates by the distance-based matching model and then fine tuning the ranking using the Transformer model. The achieved f1 Score is 44.9% which shows a 36.8% improvement over the distance-based model and a 55.36% improvement over the Transformer-based model when these models are used in isolation. It is important to notice that although the Transformer model achieved an f1 Score of 80.1% on pairwise predictions, i.e., examples of individual look-ahead planning tasks and master activities, this performance did not translate to a high matching score. This is caused by the amount of predictions the Transformer model has to perform for each look-ahead planning task where that task needs to be checked against all the available master activities which decreases the probability of the model picking the exact correct match [12]. Introducing a multi-stage ranking process where a distance-based model generates a list of candidates for the Transformer model improves the results because it mostly eliminates wrong candidates and reduces the overall number of judgements required by the Transformer model.

Although the Transformer model presented significant improvements over the distance-based matching model, an f1 Score of 44.9% can still be low for the system to be used fully automatically. To that end, we test our system's ability to fulfil a semi-automated workflow where the model returns a list of five master activities as opposed to only one. By comparing the results of the semi-automated approach (Table 6) to the results of the fully-automated approach (Table 5), one can identify a 55.4% improvement in the f1 Score which increased from 44.9% to 69.8%. While the results of the fully-automated approach suggest a need for additional enhancements, the results of the semi-automated approach are promising and show potential for being tested in production.

In regards to the generation of look-ahead plan tasks from master schedule activities, Table 10 highlights how the model accurately broke down the master activity into production tasks such as installing bathroom accessories, painting, and installing wall and floor tiles. Similarly, the last example shows task predictions associated with "*L11 Reshoring and Pour Deck & Columns*". While the model accurately generated tasks related to Rebar, PT [Post-Tension], Pouring, and top embeds which are all concrete related activities, it also falsely generated activities that are related to finishing activities instead of concrete placement activities. These results highlights the potential of modern language models such as GPT-2 [14] even when fine-tuned on relatively small datasets such as ours. The task descriptions generated by our model are comparable to those actually written by superintendents (Tables 1 and 8) in terms of sentence structure and composition as well as coherence and style.

In terms of limitations, our model learns the biases, mistakes, and particularities that exist in the training data. Like other data-driven models, those biases and irregularities are expected to be minimized as we train the model on larger datasets. Similarly, one other inherent limitation of data-driven approaches is the unpredictability of the nature

Table 10
Qualitative results showing the generative capabilities of the supervised matching model. The tasks highlighted in blue are accurate tasks generated by the model while those highlighted in red are inaccurate look-ahead planning task suggestions.

Master Activity		Model Input	Model Output: Look-ahead Planning Tasks	Correct
WBS 1 > WBS 2	Description			
construction > floor fit-outs	finishes level 4	construction <SEP> floor fit-outs <SEP> finishes level 4 <EOS>	level 04 fire protection trim out	√
			level 04 install gwb at partition	√
			level 04 install gwb at coffer ceiling	√
			level 03 ceiling coffer mep rough in	√
			fab deliver light fixture	√
Interior Build out & Apartment Finishes > Level 15	Doors & Hardware	Interior Build out & Apartment Finishes <SEP> Level 15 <SEP> Doors & Hardware <EOS>	llevel 15 interior mirror shower glass shelving	√
			level 21 interior door and trim	√
			electrical	x
			level 15 interior reswing door	√
			level 15 interior bathroom	x
Superstructure > L9	concrete deck L9	Superstructure <SEP> L9 <SEP> concrete deck L9 <EOS>	l 9 deck install table	√
			l 9 deck top embeds	√
			mechanical plumbing level 26 pre fab	x
			l 9 deck pour	√
			level 01 install ahus euhs	x
Superstructure > L8 to L36	L11 Reshoring Pour Deck & Columns	Superstructure <SEP> L8 to L36 <SEP> L11 Reshoring Pour Deck & Columns <EOS>	l 11 deck electrical	√
			l 11 deck rebar and pt deck	√
			l 11 deck pour l 11	√
			mechanical mechanical oh rough level 26	x
			l 11 deck top embeds	√

and gravity of their mistakes. To that end, this model is intended as an assistant as opposed to being a fully automated system that replaces the work done by human planners. We envision this model to be a suggestive model within a human-in-the-loop approach that assist superintendents create the look-ahead planning tasks based on the master activities created by the planners, as well as help planners identify look-ahead planning tasks related to different master schedule activities for reporting purposes. Furthermore, the presented research can be tightly linked to AI-based automated progress monitoring models. While these models focus on capturing the actual site conditions to enable progress reporting, the ScheduleAlignmentAide model presented in this manuscript can help translate the progress detected on the look-ahead plan task level into its master schedule level and vice versa by establishing and maintaining the link between the two. This enables automated progress monitoring methods to reflect progress on both short-term and long-term plan levels.

6. Conclusions

This paper presented first-of-its-kind (a) new formalization, (b) NLP solution for automatically matching look-ahead planning tasks to master schedule activities, and (c) AI solution that can generate look-ahead planning tasks from an input master activity prompt. A ranking problem was formulated where we focus on retrieving the most relevant master schedule activity for an input look-ahead planning task. We examined multiple models to the solution including combinations of distance, location, and deep learning-based ranking models and two approaches: a fully automated approach where matching is performed completely automatically, and a semi-automated approach that follows a human-in-the-loop methodology where the model suggests a list of potential matches to the human planner and the planner performs the final assignment. Our findings show that the semi-automated approach can be performed with 76.5% Precision when using a multi-stage matching model where distance-based matching is used to generate candidates to a supervised Transformer-based model which then performs the final ranking. On the other hand, the fully automated approach is still immature to be used in the industry where the best model scored 51.1% Precision. Future efforts can build on top of this work by improving the matching results through training on larger datasets, fine tuning hyper-parameters, or trying new matching models altogether. Additional efforts can also be invested in building a human-in-the-loop solution that enables a human planner to easily interact with the outputs of the model and uses the corrections made by the planner to further improve the matching results. Similarly, A/B testing with the help of a control group on an on-going construction project can help highlight the value and the immediate benefits of the proposed method. Lastly, the possibility of mapping project constraints and issues to look-ahead plans and master schedules can be investigated.

Declaration of Competing Interest

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

Acknowledgements

This material is in part based upon works supported by the National Science Foundation [1446765, 2020227]. The support and help of construction companies in collecting schedule data and validating the system prototype is greatly appreciated. The opinions, findings, and conclusions or recommendations expressed are those of the authors and do not reflect the views of the NSF, or the companies mentioned above.

References

[1] F.R. Hamzeh, G. El Samad, S. Emdanat, Advanced metrics for construction planning, J. Construct. Eng. Manage. 145 (11) (2019) 04019063, https://doi.org/10.1061/(ASCE)CO.1943-7862.0001702.
[2] D.W. Halpin, B.A. Senior, Construction Management, John Wiley & Sons, 2011. ISBN: 978-1-119-25680-9.
[3] H.G. Ballard, The Last Planner System of Production Control, University of Birmingham, 2000. https://leanconstruction.org/uploads/wp/media/docs/ballard2000-dissertation.pdf (visited 21.08.21) (Ph.D. Thesis).
[4] R. Sacks, O. Sepp“anen, V. Priven, J. Savosnick, Construction flow index: a metric of production flow quality in construction, Construct. Manage. Econ. 35 (1–2) (2017) 45–63, https://doi.org/10.1080/01446193.2016.1274417.
[5] F.R. Hamzeh, G. Ballard, I.D. Tommelein, Improving construction workflow-the connective role of lookahead planning, in: Proceedings for the 16th Annual Conference of the International Group for Lean Construction, 2008, pp. 635–646, https://doi.org/10.13140/RG.2.1.3804.3685.
[6] F. Hamzeh, G. Ballard, I. Tommelein, Rethinking lookahead planning to optimize construction workflow, Lean Construct. J. (2012) 15–34.
[7] G. Ballard, G.A. Howell, An update on last planner, in: 11th Annual Conference of the International Group for Lean Construction, Virginia, USA, 2003.
[8] J.L. Fernandez-Solis, V. Porwal, S. Lavy, A. Shafaat, Z.K. Rybkowski, K. Son, N. Lagoo, Survey of motivations, benefits, and implementation challenges of last planner system users, J. Construct. Eng. Manage. 139 (4) (2013) 354–360, https://doi.org/10.1061/(ASCE)CO.1943-7862.0000606.

[9] J.J. Lin, M. Golparvar-Fard, Visual data and predictive analytics for proactive project controls on construction sites, in: Domer B., I. Smith (Eds.), Advanced Computing Strategies for Engineering – 25th EG-ICE International Workshop 2018, Proceedings, vol. 10863 of Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), Springer, Germany, 2018, pp. 412–430, https://doi.org/10.1007/978-3-319-91635-4_21.

[10] S. Emdanat, M. Azambuja, Aligning near and long term planning for LPS implementations: a review of existing and new metrics, Lean Construct. J. (2016) 90–101.

[11] B. Dave, J.-P. H”am“al”a inen, L. Koskela, et al., Exploring the recurrent problems in the last planner implementation on construction projects, in: Proceedings of the Indian Lean Construction Conference, Institute for Lean Construction Excellence, Mumbai, India, 2015, p. 9. http://urn.fi/URN:NBN:fi:aalto-201503031948.

[12] J.J. Lin, M. Golparvar-Fard, Construction progress monitoring using cyber-physical systems, in: C.J. Anumba, N. Roofigari-Esfahan (Eds.), Cyber-Physical Systems in the Built Environment, Springer, 2020, pp. 63–87, https://doi.org/10.1007/978-3-030-41560-0_5.

[13] J.J. Lin, M. Golparvar-Fard, Visual and virtual production management system for proactive project controls, J. Construct. Eng. Manage. 147 (7) (2021) 04021058, https://doi.org/10.1061/(ASCE)CO.1943-7862.0002045.

[14] X. Wu, M. Lode, Language models are unsupervised multitask learners, OpenAI Blog 1 (8) (2019). http://www.persagen.com/files/misc/radford2019language.pdf (visited 21.08.21).

[15] B. Dave, R. Sacks, Production control systems for construction at the nexus of lean and BIM, in: Lean Construction: Core Concepts and New Frontiers, Routledge, 2020, pp. 54–84, https://doi.org/10.1201/9780429203732.

[16] G. Ballard, I. Tommelein, Current process benchmark for the last planner system, Lean Construct. J. (2016) 57–89.

[17] R. Sacks, What constitutes good production flow in construction? Construct. Manage. Econ. 34 (9) (2016) 641–656, https://doi.org/10.1080/01446193.2016.1200733.

[18] D. Heigermoser, B.G. de Soto, E.L.S. Abbott, D.K.H. Chua, BIM-based last planner system tool for improving construction project management, Autom. Construct. 104 (2019) 246–254, https://doi.org/10.1016/j.autcon.2019.03.019.

[19] F. Amer, M. Golparvar-Fard, Modeling dynamic construction work template from existing scheduling records via sequential machine learning, Adv. Eng. Inform. 47 (2021) 101198, https://doi.org/10.1016/j.aei.2020.101198.

[20] M.A. Fischer, F. Aalami, Scheduling with computer-interpretable construction method models, J. Construct. Eng. Manage. 122 (4) (1996) 337–347, https://doi.org/10.1061/(ASCE)0733-9364(1996)122:4(337.

[21] C. Hendrickson, C. Zozaya-Gorostiza, D. Rehak, E. Baracco-Miller, P. Lim, Expert system for construction planning, J. Comput. Civ. Eng. 1 (4) (1987) 253–269, https://doi.org/10.1061/(ASCE)0887-3801(1987)1:4(253.

[22] A. Darwiche, R.E. Levitt, B. Hayes-Roth, OARPLAN: generating project plans by reasoning about objects, actions and resources, Artif. Intell. Eng. Des. Anal. Manuf. 2 (3) (1988) 169–181, https://doi.org/10.1017/S0890060400000639.

[23] F. Amer, H.Y. Koh, M. Golparvar-Fard, Automated methods and systems for construction planning and scheduling: critical review of three decades of research, J. Construct. Eng. Manage. 147 (7) (2021), https://doi.org/10.1061/(ASCE)CO.1943-7862.0002093.

[24] F. Amer, M. Golparvar-Fard, Automatic understanding of construction schedules: part-of-activity tagging, in: Proceedings of the 2019 European Conference for Computing in Construction, vol. 1, 2019, pp. 190–197, https://doi.org/10.35490/ec3.2019.196.

[25] H. Alikhani, C. Le, H.D. Jeong, A deep learning algorithms to generate activity sequences using historical as-built schedule data, in: Creative Construction e-Conference 2020, Budapest University of Technology and Economics, 2020, pp. 2–6, https://doi.org/10.3311/CCC2020-039.

[26] X. Zhao, K.-W. Yeoh, D.K.H. Chua, Extracting construction knowledge from project schedules using natural language processing, in: 10th International Conference on Engineering, Project, and Production Management, Springer, 2020, pp. 197–211, https://doi.org/10.1007/978-981-15-1910-9_17.

[27] M. Bilal, L.O. Oyedele, J. Qadir, K. Munir, S.O. Ajayi, O.O. Akinade, H.A. Owolabi, H.A. Alaka, M. Pasha, Big data in the construction industry: a review of present status, opportunities, and future trends, Adv. Eng. Inform. 30 (3) (2016) 500–521, https://doi.org/10.1016/j.aei.2016.07.001.

[28] J. Zhang, N.M. El-Gohary, Automated reasoning for regulatory compliance checking in the construction domain, in: Construction Research Congress 2014: Construction in a Global Network, 2014, pp. 907–916, https://doi.org/10.1061/9780784413517.093.

[29] P. Zhou, N. El-Gohary, Automated extraction of environmental requirements from contract specifications, in: Proceedings of the 16th International Conference on Computing in Civil and Building Engineering, vol. 1676, 2016, p. 1669.

[30] J.-R. Lin, Z.-Z. Hu, J.-P. Zhang, F.-Q. Yu, A natural-language-based approach to intelligent data retrieval and representation for cloud bim, Comput. Aided Civ. Infrastruct. Eng. 31 (1) (2016) 18–33, https://doi.org/10.1111/mice.12151.

[31] M. Kopsida, I. Brilakis, P. Vela, A review of automated construction progress monitoring and inspection methods, in: Proceedings of the 32nd International Conference of CIB W78, Eindhoven, The Netherlands, 2015 27-29 October, pp. 421–431.

[32] B. Ekanayake, J.K.-W. Wong, A.A.F. Fini, P. Smith, Computer vision-based interior construction progress monitoring: a literature review and future research directions, Autom. Construct. 127 (2021) 103705, https://doi.org/10.1016/j.autcon.2021.103705.

[33] T. Sawyer, Model. Supply Chains 260 (14) (2008) 24–27. http://worldcat.org/issn/08919526 (visited 21.08.21).

[34] P. Jafari, M. Al Hattab, E. Mohamed, S. AbouRizk, Automated extraction and time-cost prediction of contractual reporting requirements in construction using natural language processing and simulation, Appl. Sci. 11 (13) (2021), https://doi.org/10.3390/app11136188.

[35] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146.

[36] N.P. Garcia-Lopez, M. Fischer, A construction workflow model for analyzing the impact of in-project variability, in: Construction Research Congress 2016, 2016, pp. 1998–2007, https://doi.org/10.1061/9780784479827.199.

[37] P.J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, N. Shazeer, Generating Wikipedia by Summarizing Long Sequences, 2018. http://arxiv.org/abs/1801.10198 (visited 21.08.21).

[38] M.E. Peters, M. Neumann, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1 (Long Papers), Association for Computational Linguistics, New Orleans, Louisiana, 2018, pp. 2227–2237, https://doi.org/10.18653/v1/N18-1202.

[39] R. Józefowicz, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu, Exploring the Limits of Language Modeling, 2016. http://arxiv.org/abs/1602.02410 (visited 21.08.21).

[40] S. Merity, N. Shirish Keskar, R. Socher, Regularizing and Optimizing LSTM Language Models, CoRR (2017). http://arxiv.org/abs/1708.02182 (visited 21.08.21).

[41] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, in: Y. Bengio, Y. LeCun (Eds.), 1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, 2013. May 2–4, 2013, Workshop Track Proceedings.

[42] D. Bahdanau, K.H. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, in: Y. Bengio, Y. LeCun (Eds.), 3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7–9, 2015.

[43] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, CoRR (2017). http://arxiv.org/abs/1706.03762 (visited 21.08.21).

[44] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving Language Understanding by Generative Pre-Training, 2018. https://www.semanticscholar.org/paper/Improving-anguage-Understanding-by-Generative-Radford-Narasimhan/cd18800a0fe0b668a1cc19f2ec95b5003d0a5035 (visited 21.08.21).

[45] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, CoRR (2018). https://arxiv.org/abs/1810.04805 (visited 21.08.21).

[46] A. Fan, M. Lewis, Y. Dauphin, Hierarchical neural story generation, in: I. Gurevych, Y. Miyao (Eds.), Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15–20, 2018, Volume 1: Long Papers, Association for Computational Linguistics, 2018, pp. 889–898, https://doi.org/10.18653/v1/P18-1082.

[47] A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi, The Curious Case of Neural Text Degeneration, 2019. http://arxiv.org/abs/1904.09751 (visited 21.08.21).