© 2021 Fouad Amer

LEARNING FORMAL ACTIVITIES REPRESENTATION FOR AUTOMATED QUALITY CONTROL, ALIGNMENT, AND REVISION OF CONSTRUCTION SCHEDULES

BY

FOUAD AMER

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Civil Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2021

Urbana, Illinois

Doctoral Committee:

Associate Professor Mani Golparvar-Fard, Chair and Director of Research
Professor Julia Hockenmaier
Professor Khaled El-Rayes
Associate Professor Liang Liu
Associate Professor Nora El-Gohary

ABSTRACT

Construction planning requires domain know-how, which is often ingrained with experienced construction planners through exposure to multiple projects over decades of work experience. Without explicit in-depth prior knowledge of the underlying processes, planning experts learn to form empirical associations between project problems and potential solutions. Almost in all situations, their experience is described through examples from previous projects as opposed to structured and principled knowledge. Given the lack of formal definitions and structured knowledge, transferring expert human planners' comprehensive knowledge to automated planning systems proves to be complex. And, while substantial research is conducted and many automated planning systems were developed, the complexities of actual construction projects on one hand, and the nuances between different construction approaches, on the other, have rendered the manual creation of explicit knowledge representations extremely tedious and, in many cases, entirely infeasible. Hence, most –if not all– construction projects still rely on a single or a small group of experienced practitioners to manually perform their planning and scheduling activities.

This dissertation presents a suite of machine learning (ML) and natural language processing (NLP) based methods and systems to learn formal construction planning and sequencing knowledge representations from existing construction records. The learned knowledge is leveraged to (1) generate lists of potential successor activities given a sequence of predecessors, (2) retrieve lists of predecessor activities required to enable the execution of a sequence of successors, (3) critique the quality of logic of an input construction schedule by identifying missing or wrong logical dependencies, (4) schedule a set of unordered activities, (5) align weekly work plans (WWPs) to master schedules following a semi-automated human-in-the-loop approach, (6) identify the implicit dependency constraints governing predecessor-successor relationships and their roles and flexibilities in an input construction schedule, and (7) revise lookahead plans to mitigate the

effects of unavoidable activity delays. All the developed methods learn using free form activity descriptions and Work Breakdown Structure (WBS) tags only without any prior assumptions or requirements on the expressions used to represent construction activities. They are trained and tested on a dataset of 32 construction schedules collected from real projects from different industry segments and various construction companies. Quantitative and qualitative results are presented to show the validity and the potential of the proposed methods. Limitations and future work are discussed in detail.

To my hard working parents

ACKNOWLEDGMENTS

When I first joined the CEE program at UIUC, my plan was to graduate with a masters degree in construction engineering, until Mani, my advisor and mentor, shifted my perspective! He believed that the construction industry is ripe for transformation, and that transformation must come from multi-disciplinary researchers who have deep knowledge in both construction and technology. He pushed me to study artificial intelligence, natural language processing, computer vision, and machine learning. He believed that change depends on collaboration between industry and academia, and he enabled and supported me to be involved in both. Thank you Mani for your vision, your ambition, your energy, and your belief in me and the entire team. Your mentorship changed my life.

Coming from a civil engineering background, diving deep into natural language processing was a challenging task, and it would have been extremely difficult without the continuous help and support from Prof. Julia. Thank you so much Julia for your interest in my research and for the long hours you spent with me helping me think through my problems, sending me articles to read, and reviewing my results.

Doing interdisciplinary technical research at the intersection of artificial intelligence and construction requires spending long hours writing and debugging code, training and testing models, and getting lost in data analysis. It's easy to get lost in the details and lose track of what's important from an application perspective. For that, Professors Khaled, Liu, and Nora helped me wear my hard hat and re-evaluate my research from a construction perspective. Thank you all so much for your rigorous reviews and genuine comments.

Spending long hours in Newmark would have been impossible without my RAAMAC family: Jacob, Juan, Wil, Amir, Shuai, Dominic, Yuna,and Yeritza this wouldn't have been possible without you. Thank you for all the time we spent together learning new things, arguing, laughing, and most

importantly complaining! (just kidding!)

My parents, brother, sisters, nephews, and nieces, your hearty and warm welcomes when I go back home are what keeps me going. I can't thank you enough for all your calls, all the jokes you send, your endless unconditional support, and your insistence on sharing your life with me although I am miles and miles away.

Nour, my partner in crime, thank you for opening up my eyes to new opportunities and encouraging me to take risks, for being there for me and sticking around all this time, for hours and hours of Skype calls, and for never missing an important event in my life!

Last but not least, those who lived in Urbana-Champaign (or Champoo-Banana as we like to call it) know that it's all about community. I am ever grateful for my roommates and all my friends for always being there for me and for sharing my daily life.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 AUTOMATED METHODS AND SYSTEMS FOR CONSTRUCTION PLANNING AND SCHEDULING: A CRITICAL REVIEW ON THREE DECADES OF RESEARCH [1] 9

CHAPTER 3 MODELING DYNAMIC CONSTRUCTION WORK TEMPLATE FROM EXISTING SCHEDULING RECORDS VIA SEQUENTIAL MACHINE LEARNING [2] 39

CHAPTER 4 LEARNING AND CRITIQUING PAIRWISE ACTIVITY RELATIONSHIPS FOR SCHEDULE QUALITY CONTROL VIA DEEP LEARNING-BASED NATURAL LANGUAGE PROCESSING [3] 75

CHAPTER 5 TRANSFORMER MACHINE LEARNING LANGUAGE MODEL FOR AUTO-ALIGNMENT OF LONG-TERM AND SHORT-TERM PLANS IN CONSTRUCTION [4] 118

CHAPTER 6 CONSTRUCTION SCHEDULE AUGMENTATION WITH IMPLICIT DEPENDENCY CONSTRAINTS AND AUTOMATED GENERATION OF LOOKAHEAD PLAN REVISIONS 153

CHAPTER 7 CONCLUSIONS AND FUTURE WORK 177

REFERENCES 186

CHAPTER 1

INTRODUCTION

Planning for any new project requires a clear definition of the project scope. Once the scope is well defined, it is further broken down into a hierarchy of manageable work packages formally known as a Work-Breakdown-Structure (WBS) [1, 2]. The WBS defines the work required to deliver the project, and it forms the basis for designing the construction process. Given a well-defined WBS, planners schedule each task by estimating duration requirements and accounting for sequencing dependencies governed by flows of precedence, resource requirements (i.e., crews, equipment, and material), information (e.g., Request for Information (RFI) and task constraints in lean projects) and workspaces [3]. The challenge is that such estimations require domain know-how, which is often ingrained with experienced construction planners through exposure to multiple projects over decades of work experience. Without explicit in-depth prior knowledge of the underlying processes, planning experts learn to form empirical associations between project problems and potential solutions. Almost in all situations, their experience is described through examples from previous projects as opposed to structured and principled knowledge. They often know more than what they can explain, and their know-how proves hard to transfer to junior planners [2]. Given the lack of formal definitions and structured knowledge, transferring expert human planners' comprehensive knowledge to automated planning systems proves to be complex.

While substantial research is conducted and many automated planning systems were developed, the feasibility of scaling these systems to real-world projects is not fully examined. Chapter 2 dives into the literature of automated planning and scheduling to understand the reasons hindering the broad adoption of such systems. The complexities of actual construction projects on one hand, and the nuances between different construction approaches, on the other, have rendered the manual creation of explicit knowledge representations extremely tedious and, in many cases, entirely infeasible. Hence, most –if not all– construction projects still rely on a single or a small group of

experienced practitioners to manually perform their planning and scheduling activities. Current project-specific approaches do not benefit from years of experience accumulated by other planners in similar situations and, in turn, can not guarantee the optimality of generated plans. Given the availability of large-scale historical data, machine learning poses a potential solution to tackle the scalability issue associated with capturing and sharing planning and scheduling knowledge. As such, we explore the hypothesis that *planning and scheduling knowledge and work templates can be modeled using machine learning, with minimal human input*. The need for accumulated experiences from previous projects, the availability of planning and sequencing knowledge "big" data in the form of master schedules and WWPs, and the leaps in machine learning (ML) and natural language processing (NLP) research capable of leveraging historical records offer a unique opportunity to the construction industry. We leverage this opportunity to fulfill the following research objectives:

1.1 Research Objectives

This research aims at creating and validating a suite of Machine Learning-based methods for a system that (1) learns best practices of construction planning and scheduling at the levels of (a) master activities and (b) weekly tasks, (2) offers schedule quality control, and (3) provides prescriptive solutions in the form of revised lookahead plans.

Objective 1.Formalize and validate construction planning knowledge that enables automation of construction planning and learning from existing records. The automation of construction planning and scheduling grabbed the attention of many researchers since the early 80s. [4, 5, 6, 7, 8, 9, 10, 11] are few examples. However, previously developed approaches relied heavily on manual process modeling and explicit knowledge representation. Nowadays, given the availability of “big” construction planning and scheduling data in the form of previous schedules and the advancements in machine learning, natural language processing, and data mining research, the construction industry has a unique first-time opportunity of harnessing construction insights automatically from hundreds of previous projects. The first step towards that goal, and the first objective of this thesis, is to formalize construction planning and sequencing knowledge in flexible schemas that enable

the application of machine learning and NLP techniques.

Objective 2. Create and validate a new method for mining best practices in construction planning from existing company-specific records. After formalizing construction planning and scheduling knowledge in flexible data schemes that enable learning from existing records, this research aims at developing and validating a new machine learning-based method for mining construction planning knowledge encoded in existing company records. Given the sequential nature of construction schedules, sequential machine learning models such as Long Short Term Memory Recurrent Neural Networks [12] and more recent attention-based architectures [13] will be investigated. This objective is tightly coupled with objective 1, and they will be investigated simultaneously. Similarly, both objectives work as the basis for the remaining 3 objectives.

Objective 3. Create and validate a new method to evaluate quality of construction schedules in terms of sequencing logic. High quality construction schedules provide construction projects with a tool to improve construction quality and productivity through improving the control process [14]. To evaluate and critique the quality of construction schedules, [15, 16, 17, 18, 19, 20] presented multiple scoring guidelines which all included the correctness of critical path and sequencing logic. Previous attempts to automate the process of critiquing the quality of schedule logic –such as [15, 21, 17, 16] among others– used generic activities, hard-coded sequencing rules, and/or manually set scoring thresholds based on interviews with industry experts. A method that automatically learns construction sequencing knowledge from existing historical records and applies it for the purpose of sequencing logic critiquing has not been pursuit yet. In this research, the learned generative models representing the best practices of a specific company –built in Objective 1– can be leveraged to help planners judge the quality of a new schedule. Master schedules often contain explicit precedence relationships. We hypothesize that given enough schedules, an AI will learn what sequences of activities are likely to be correct, and what sequences are unlikely to be produced by the learned model. Such AI will enable us to flag potential sequencing logical flaws and to inform project management teams of potential risks of delays.

Objective 4: Create and validate a new human-in-the-loop method to semi-automatically align weekly tasks to master schedule activities. Construction practitioners struggle aligning short-term to long-term plans, and they invest significant efforts to maintain such alignments. The time required to synchronize the WWPs, look-aheads, and master schedules often exceeds the planning cycle time [22, 23]. More companies are currently using a combination of CPM-based long-term master schedules to control overall project performance and LPS-based WWPs to manage detailed daily tasks. While the master schedule is built by company planners who are not always allocated to a specific project, WWPs are built by superintendents and project teams. The language used for master schedules and WWPs in terms of activity names, granularities, relationships, and correspondences is not the same. The decoupling between short- and long-term plans practically starts here. Owners request monthly updates on the master schedule which is controlled for performance through EVA methodologies and metrics such as Schedule and Cost Performance Indices (SPI and CPI respectively). On the other hand, WWPs are mostly controlled for plan reliability and learning through LPS-based measures such as PPC, Task Anticipated, etc. This research targets this gap. We aim at enabling the measurement of WWP performance through aligning it to the master schedule using NLP techniques. Given a set of weekly tasks, we identify the set of master schedule activities that corresponds to these tasks semi-automatically to enable a continuous online update of the master schedule from individual weekly tasks.

Objective 5: Develop a learning-based system that generates revised optimal look-ahead plans to help project teams improve the basis of project controls and mitigate unavoidable activity delays. Construction researchers devised multiple metrics for proactively identifying locations and activities which are at risk of delay [24, 25, 26, 27, 28]. Researchers also examined many schedule generation and optimization techniques [29, 30, 31, 6, 10, 32, 33, 11]. To the best of our knowledge, a model that proactively identifies risky activities and offers optimal look-ahead schedule revisions based on best practices learned from previous data has not been explored yet. The approaches that generate optimal schedules rely on manually predefined activities and/or manually elaborated recipes that describe activities and sequencing logic. In this research, we expand the capabilities of DPTs– described in Objective 1– to enable the generation of revised optimal lookahead plans to mitigate unexpected activity delays. This prescriptive approach augments the

capabilities of project teams by offering actionable insights learned from construction scheduling data which is accumulated over hundreds of previous projects.

1.1.1 Thesis Organization

Chapter 2 offers a critical review of the literature. It examines the key issues that, to date, have hindered scaling and wide adoption of automated planning methods and systems. Over the past three decades, a large body of research focused on automated construction planning and scheduling. Some of these efforts introduced methods to use design information to automatically develop the scope of work, establish work breakdown structures, and create optimal project sequences. Others introduced new techniques to formalize the sequencing relationships among schedule activities and project components. Despite these advancements, most construction projects –if not all– are still fully engaged in manual workflows of planning and scheduling. A close examination on how knowledge is formalized; scope quantification and project definition methods; and planning, scheduling, and schedule optimization techniques has identified the following gaps in knowledge: (1) lack of flexibility in how construction knowledge is stored in existing construction method model templates for sequencing algorithms; (2) the dependency of current automated scheduling methods on manually formed and maintained work templates; (3) lack of learning methods to automate learning of construction knowledge from existing records without extensive human input; (4) limited validation on applicability of existing automated planning systems on real-life construction projects; and (5) the decoupled nature of research on automated planning vs. schedule optimization. Building on the recent advancements in deep learning and natural language processing and the rise in adoption of lean construction theories, a discussion is offered on the path for research towards automatic generation of dynamic work templates and their inclusion in integrated planning, scheduling and optimization systems.

Chapter 3 aims at fulfilling these needs via a new method to automatically learn construction knowledge from historical project planning and scheduling records and digitize such knowledge in a flexible and generalizable data schema. Today, construction planning and scheduling is almost always performed manually, by experienced practitioners. The knowledge of those individuals is materialized, maintained, and propagated through master schedules and look-ahead plans. While

historical project schedules are available, manually mining their embedded knowledge to create generic work templates for future projects or revising look-ahead schedules is very difficult, time-consuming and error-prone. The rigid work templates from prior research are also not scalable to cover the inter and intra-class variability in historical schedule activities. We present Dynamic Process Templates (DPTs) based on a novel vector representation for construction activities where the sequencing knowledge is modeled with generative Long Short-Term Memory Recurrent Neural Networks (LSTM-RNNs). Our machine learning models are exhaustively tested and validated on a diverse dataset of 32 schedules obtained from real-world projects. The experimental results show our method is capable of learning planning and sequencing knowledge at high accuracy across different projects. The benefits for automated project planning and scheduling, schedule quality control, and automated generation of project look-aheads are discussed in detail.

Chapter 4 offers a novel machine learning-based solution that learns construction scheduling domain knowledge from existing records completely automatically and applies it to validate the logic in input schedules. The success of a construction project relies on the knowledge and skills of the project planners, and how they address the challenges they face. The sheer volume of data needed in forms of project scope –usually made available though quantity take-off– and the multitude of the dependencies among activities in forms of structural, functional, legal, resource, information constraints, make planning large projects a difficult undertaking. When schedule logic and its impact on the project duration is considered, it is easy to see how small human errors can lead to direful consequences. All too often, mistakes are made by having too many constraints, neglecting trade logic or stacking trades with too many start-to-start and finish-to-finish relationships. In cases where logic is missing, the schedule has a lot of float embedded in it, which usually results in delays going undetected until late in the project execution phase. The labor-intensive and experience-driven nature of planning can easily impact the quality of a project schedule. Many project-specific conditions can be easily missed. This problem is further exacerbated when the new norm of planning is considered where construction companies put one of a few planners in charge of creating schedules for 100s of projects across their company. We aim at solving this problem using machine learning and NLP and we build an intelligent logic checking model achieving an F1 score of 88.3%. Furthermore, we tailor our method to use the learned knowledge to schedule a list of unordered activities. The details of the method, experimental

results, benefits, and limitations are discussed. In construction, schedule mistakes causing delays beyond substantial completion dates cost contractors expensive liquidated damages. Hence, several industry guidelines, such as the DCMA's 14 point assessment, define schedule quality and offer systematic methods for ensuring it. These guidelines list "logic" as an essential control metric, and they require planners to ensure their schedules are free of missing or wrong logical dependencies. Checking the logic requires extensive construction domain knowledge, and planners perform it entirely manually as there are no available software solutions that support it.

Chapter 5 presents the first attempt to automate linking look-ahead planning tasks to master-schedule activities following an NLP-based multi-stage ranking formulation. On construction sites, 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). The level of detail of the long-term plan (master schedule) varies, but this schedule is not ready to be used for production in most cases. Superintendents manage the day-to-day operations by creating mid-term (look-ahead plans) and short-term plans (Weekly Work Plans (WWPs)). While master schedules reside in standard construction planning solutions (e.g., Oracle Primavera), WWPs and look-ahead plans containing progress information live in spreadsheets or are printed on post-it notes on the trailer walls. Since the contractor is required to submit a monthly progress update to the owner, progress from WWPs must be translated to progress on the master schedule. To do so, project engineers must align short and long-term plans monthly following a rigorous manual process. This alignment proves to be a struggle for project teams [23]. However, their full-alignment is essential for project coordination, progress updating, and payment application reviews, and its absence may lead to costly litigation. Our model employs distance-based matching for candidate generation and a Transformer architecture for final matching. 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.

Revising lookahead plans is practiced in all construction projects to meet contractual deadlines, mitigate activity delays, and disentangle process bottlenecks. **Chapter 6** presents a new machine

learning-based method –which we call Implicit Logic Checker (ILC)– to learn the implicit dependency constraints and the flexibility of construction schedule relationships using a Transformer language model architecture. We then leverage relationships' flexibilities to provide a basis for creating optimal lookahead plan revisions that can mitigate the effect of unavoidable activity delays. We then deploy a Constrained Conditional (CC) model that applies learned construction planning knowledge to build revised lookahead plans and optimizes them for consistency and duration. Our ILC model is trained and tested on 35,332 manually labeled predecessor-successor relationships from eight real construction projects achieving an F1-Score of 91%. Similarly, the revised lookahead plans generated by the CC model on two case studies showed reduced overall plan durations.

Lastly, **Chapter 7** summarizes the contributions of this dissertation. It discusses the conclusions and limitations and highlights future research opportunities.

CHAPTER 2

AUTOMATED METHODS AND SYSTEMS FOR CONSTRUCTION PLANNING AND SCHEDULING: A CRITICAL REVIEW ON THREE DECADES OF RESEARCH [1]

[1]This chapter is published in the *Journal of Construction Engineering and Management* [34]:
" F. Amer, H. Y. Koh, and M. Golparvar-Fard, "Automated Methods and Systems for Construction Planning and Scheduling: Critical Review of Three Decades of Research", *Journal of Construction Engineering and Management, vol.147, no. 7*, Jul 2021, https://doi.org/10.1061/(ASCE)CO.1943-7862.0002093"

2.1 Introduction

Creating automated planning systems for construction requires transferring planning knowledge from human planners to computers using computer-interpretable representations. Such representations –referred to as construction method model templates, work templates, fragnets, or even recipes in various literature– need to be comprehensive to account for the complexities that govern the planning and execution of construction projects. They also need to generalize to the wide-range of projects across all sectors of the industry. Adaptability is needed so that work templates can accommodate for various construction techniques and project management methods. Since the early 1980s, construction researchers identified multiple values for automating the construction planning task, such as potential cuts in project duration, cost savings, and productivity improvements in project planning time and effort [35, 36]. Many automated planning systems have been modeled and deployed since then. Yet, to date, most projects in the construction industry–if not all– are still fully engaged in manual planning and scheduling workflows.

Planning for any new project requires a clear definition of the project scope. Once the scope is well defined, it is further broken down into a hierarchy of manageable work packages formally known as a Work-Breakdown-Structure (WBS) [1, 2]. The WBS defines the work required to deliver the project, and it forms the basis for designing the construction process. Given a well-defined WBS, planners schedule each task by estimating duration requirements and accounting

for sequencing dependencies governed by flows of precedence, resource requirements (i.e., crews, equipment, and material), information (e.g., Request for Information (RFI) and task constraints in lean projects) and workspaces [3]. The challenge is that such estimations require domain know-how, which is often ingrained with experienced construction planners through exposure to multiple projects over decades of work experience. Without explicit in-depth prior knowledge of the underlying processes, planning experts learn to form empirical associations between project problems and potential solutions. Almost in all situations, their experience is described through examples from previous projects as opposed to structured and principled knowledge. They often know more than what they can explain, and their know-how proves hard to transfer to junior planners [2]. Given the lack of formal definitions and structured knowledge, transferring expert human planners' comprehensive knowledge to automated planning systems proves to be complex.

The earliest attempts to automate construction planning and scheduling employed Knowledge-Based Expert-Systems (KBS) in the form of rule-based reasoning and decision-making methods that are built on top of artificial intelligence algorithms to solve complex problems [37], object-oriented programming paradigms, and process simulations to model and analyze construction operations. To date, several practical challenges have hindered scaling and wide adoption of automated planning systems:

Formal representation of work templates– Every single system in the literature requires the manual creation of knowledge bases that explicitly model construction operations. For modeling these representations –which are often called construction method model templates, work templates, fragnets, or even recipes in various literature– to be useful to practitioners, every single operation needs to be modeled. This modeling should be done at several levels of granularity to support the different planning phases. For example, creating Weekly Work Plans (WWPs) and 6-week look-ahead plans require more details compared to master schedules. Given the large variety of construction operations and the specifics that make every construction project unique, creating such work templates is an extremely challenging task.

Automation to structure and reuse best practices from previous projects– The systems introduced in the literature do not automatically structure and reuse best practice knowledge captured from previous projects. Any change to the knowledge bases in the current methods needs to be introduced manually. Attempts have been made to introduce Case-Based Reasoning (CBR)

methods to retain and reuse best practice knowledge through manually created project features. These systems retain knowledge by recording every planning case as a problem and a recommended solution. When faced with a new planning problem, the system evaluates the similarity between the new case and the cases stored in its database. If the system finds similarity, then it adapts its solution to the new problem and presents it to the user. The system also retains the new problem and the new solution as a new case for future reference [38, 39]. Nonetheless, (1) identifying features and similarity measures to describe a planning scenario and (2) designing an algorithm that automatically adapts the retrieved scenario to fit the new case are complicated tasks that still require manual tweaks by human users.

Rigidity of work templates– Automated planning systems need to scale to all types of projects; nevertheless hard-coded and manually structured work templates do not account for all construction operations, the complex interactions between different project components, and inter and intra-class variability in both construction operations and project components.

Despite the advancements in planning and scheduling over the past several decades, today, 98% of mega projects in the construction industry still incur 30% cost overruns, and 77% suffer from schedule delays that are 40% behind project baselines [40]. The *existing planning and project control solutions* still require a substantial amount of manual work for creating and maintaining their knowledge bases. The authors hypothesize that a key underlying reason is the technical limitations on how the knowledge is formally represented. There is a need for new automated techniques to create work templates that (a) match the scale and complexity of real-life construction projects; (b) can be adopted to learn from new observations (both previous and ongoing projects); and when necessary (c) can be adapted to operation or project-specific conditions. This paper scrapes through the literature to validate this hypothesis.

In the following section, the methodology of identifying and reviewing all literature and the specific research questions is introduced first. Then, the authors investigate the reasons that hindered the scalability of the developed systems by looking into the technical details of methods for (1) scope quantification and project definition, (2) planning, scheduling, and optimization, and (3) knowledge and best-practice formalization and reuse. Next, the authors offer a road map for improvement based on the advancements in deep learning and natural language processing techniques. Finally, the authors also discuss the role of the next generation of automated construction planning systems

and their synergies with lean construction theories.

2.2 Methodology

2.2.1 Scope Definition

This manuscript offers a thorough examination of the existing literature to highlight the knowledge gaps that have hindered the wide adoption of automated construction planning and scheduling solutions. First, in this paper, similar to [41], the term ***planning and scheduling*** describes the scope and time-related aspects of the planning and scheduling process, which is defined by a mathematical set (see Eq. 2.1). As such, planning activities associated with cost estimation, uncertainty quantification, or resource estimation and allocation as shown in Muñoz-Avila et al. [1] are not covered.

$$Planning\ \&\ Scheduling = Defining\{\ WBS,\ Work\ Activities,\ Work\ Logic\} \tag{2.1}$$

Second, an important distinction is drawn between automated planning and scheduling vs. schedule optimization methods. Although sequencing is a fundamental aspect of planning and scheduling, this research is not specifically targeted at systems where the main focus is multi-objective schedule optimization, resource leveling, or activity crashing. These systems usually do not have a knowledge base, and they are explicitly provided with a list of planning activities and the logical dependencies that govern them. Because the hypothesis presented in this research is particularly focused on the limitations of knowledge representation, the authors focus particularly on systems that can automatically (1) define the required scope of work, (2) reason about sequencing the activities, or (3) automatically perform work generation and sequence tasks. While schedule optimization is considered as a criterion, only those systems that explicitly model construction knowledge and know-how to reason about schedule generation and sequence work activities are considered in this literature review. In other words, this research includes the entire literature related to automated planning that contains multi-objective optimization, however it excludes any multi-

objective optimization research that does not offer any automated planning methods. Third, this research excludes previous efforts related to estimating activity duration, productivity forecasting, or learning curve analysis. Fourth, this work focuses on planning the construction process of the physical project components, i.e., it excludes literature related to site logistics planning, site layout planning, material handling, crane operations, or equipment path planning. Fifth, this paper focuses on construction operations that happen on-site while following conventional construction means and methods. As such, this review excludes research on 3D printing, precast components production and assembly, off-site construction, or smart factories. Last, many construction topics intertwine with automated construction planning and scheduling, and as the authors clarify in the next section, searching literature databases will return many irrelevant hits. To focus this literature review and to offer insight for others to build upon it, the authors explicitly define their exclusion criteria to be the following:

- Multi-objective schedule optimization.
- Resource leveling or activity crashing.
- Estimating activity duration, productivity forecasting, or learning curve (productivity) analysis.
- Site logistics or layout planning, material handling, crane operations, or equipment/robotic path planning.
- 3D printing, precast components production and assembly, off-site construction, or smart factories.
- Automated and/or model-driven progress tracking.
- Design and engineering, infrastructure management and prioritization, resilient systems, urban utilities, transportation systems, or railways.
- Project financing or economic evaluation.
- Non-construction related planning and scheduling (e.g., space missions planning).

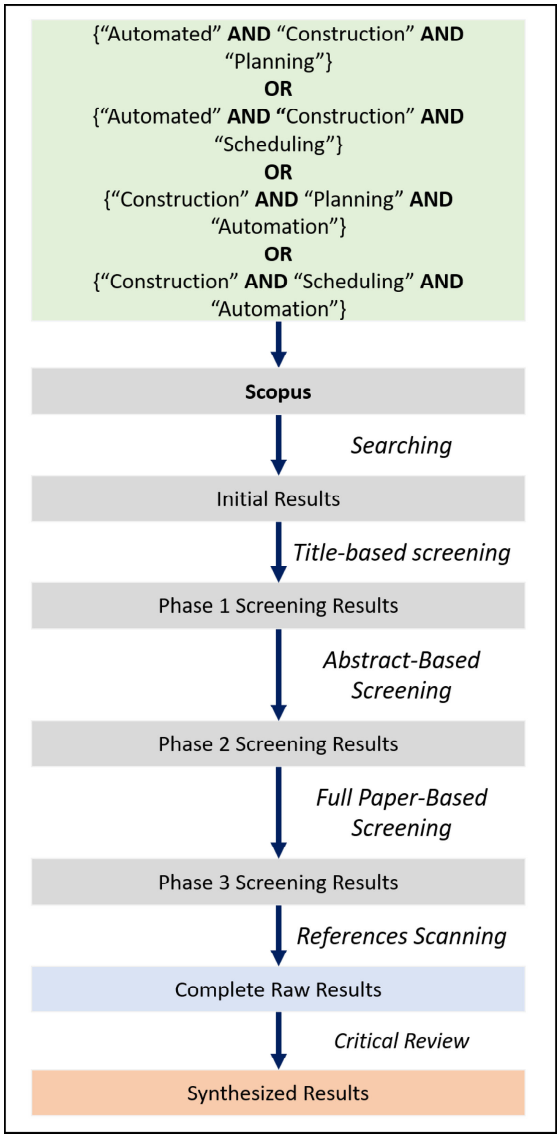


Fig. 2.1. Methodology for identifying relevant research literature.

The authors chose **Scopus** to search for the relevant literature. The search was conducted using the following search query {"Automated" **AND** "Construction" **AND** "Planning"} **OR** {"Automated" **AND** "Construction" **AND** "Scheduling"} **OR** {"Construction" **AND** "Scheduling" **AND** "Automation"} **OR** {"Construction" **AND** "Planning" **AND** "Automation"}. The initial search results included **1,565** articles. Following the review methodology highlighted in Fig.3.7 and common methodologies used in the literature, such as Sepasgozar et al. [42] and Charef et al. [43], the initial results were filtered and refined over four phases: The first phase examined article titles, which resulted in a total of **203** filtered publications. The second phase considered abstracts, which resulted in a total of **104** articles. The third phase investigated the entire body of each manuscript, maintaining a total of **26** articles. Using these steps, the results from Scopus left out many essential articles. In an attempt to ensure the completeness of the literature review, the authors performed a fourth phase of refinement to the results by examining the references of each article included in the review following the same three-phase methodology. This process appended an extra **67** manuscripts to the list of relevant articles raising the total to **93** papers. These papers –shown in Tables.2.1, 2.2, and 2.3– form the basis of all statistics, discussions, and conclusions highlighted in the following sections. In these tables, each item represents the capacity of the examined method or system against the questions and criteria that were explained above. To better understand how knowledge was disseminated, an examination was performed on venues where relevant work was published. Fig.2.2 shows the distribution of publications over their sources. 52.6% of the publications that have contributed most significantly to the topic are journal articles published in Journal of Automation in Construction, Journal of Computing in Civil Engineering, and Journal of Construction Engineering and Management respectively. Other publication sources consist of peer-reviewed conferences, theses, technical reports, and book chapters.

2.2.2 Research Questions

The authors hypothesize that a key underlying technical reason that has widely limited the adoption of automated planning systems in the industry is the substantial amount of manual work involved with creating and maintaining their knowledge bases. The rigid nature of the existing work templates has limited scalability of creating and maintaining automated planning systems. To

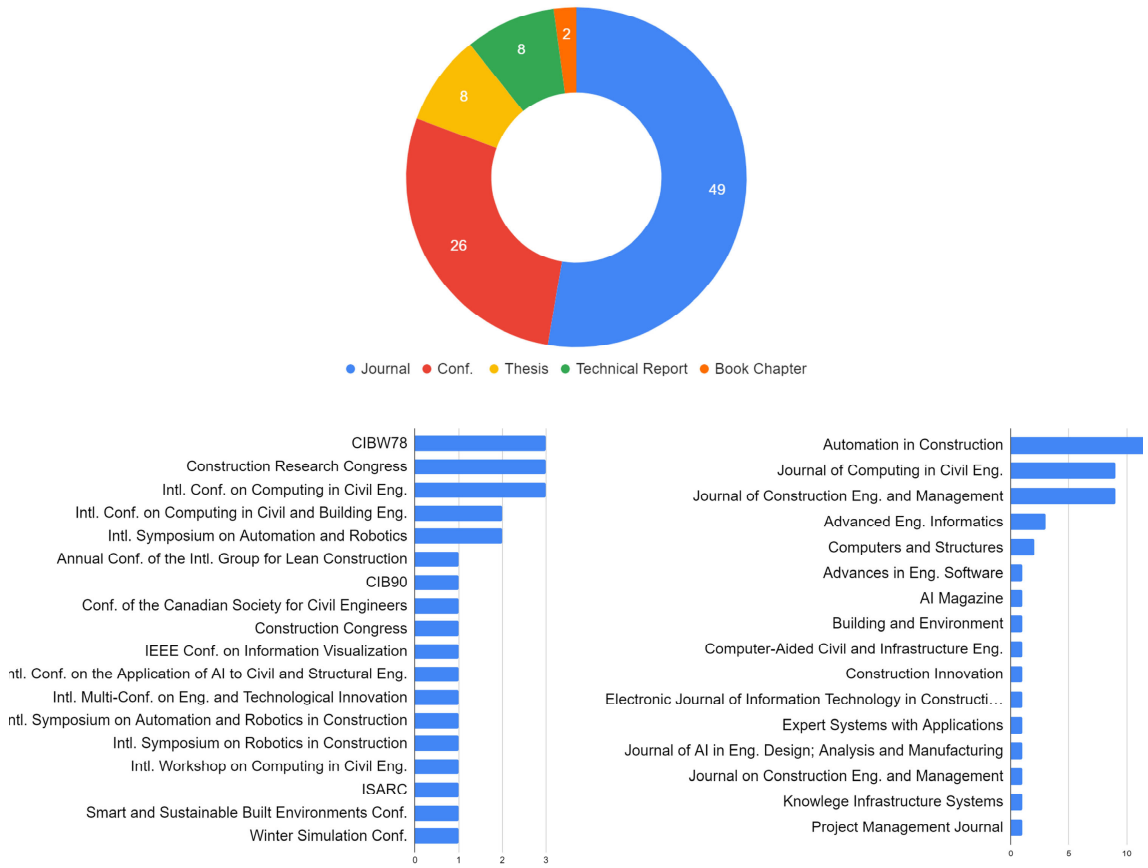


Fig. 2.2. The distribution of the published literature around automated planning and scheduling.

validate this hypothesis, this manuscript thoroughly scrapes through the construction planning and scheduling literature with the methodology described above. Specifically, to better understand the root-causes for the slow adoption of automated planning systems, the following questions are raised and responded to, per reviewed method or system:

1. How is the knowledge of construction planning and scheduling formalized, modeled, and represented?

2. How is Knowledge Base in this planning and scheduling engine populated with relevant data? Is the population of such data driven by the user (i.e., manual) or is it learned automatically?

3. Are there any procedures to continuously learn from ongoing projects to create an improved knowledge base for future projects?

The authors also investigated the feasibility of manually creating and managing rigid knowledge bases and their appeal to practitioners. To do so, key performance indicators (cost and time savings) associated with existing planning systems were investigated in terms of (a) their ability to plan real-world projects, (b) reducing planning efforts by automatically reasoning about the scope of a project using a product model (e.g., Building Information Models), or (c) optimizing a schedule. Hence, the following questions were also considered:

4. Does this method or system only consider generic construction operations or is it only applicable to project or operation-specific scenarios?

5. How is the scope of a project quantified in this method or system? Can it automatically reason about the required scope of work from a product model?

6. Does this method or system automatically perform the tasks of planning and scheduling? Is it capable of identifying the required activities for a given scope of work, or is it only focused on scheduling the manually-defined activities? Does it automatically optimize schedule sequences or activity logic?

These criteria and questions were considered in the review of each and every method or system introduced in the literature. The following sections offer a thorough discussion on the findings of this review along with recommendations for future research efforts.

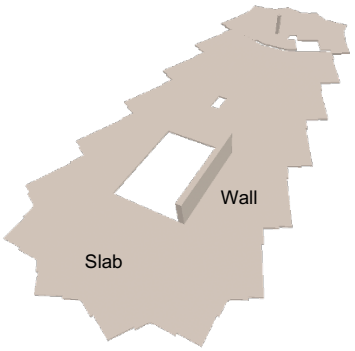
Process Models				Product Model
Activity Representation Schema		**Activity 1**	**Activity 2**	
Semantic Object Oriented	**Component**	Wall*	Slab*	
	Action	Place*	Form*	
	Resource	C3*	C1*	
	Sequencing Constraint	``Wall Formwork and Reinforcement Complete"	"Slab Supports Placed"	
Object Oriented	**Description**	Place Wall*	Form Slab*	
	Sequencing Constraint	Precedence links to "Form Wall" and "Reinforce Wall"	Precedence links to "Place Wall"	
Functional Operators	**Operator**	PlaceWall()	FormSlab()	
	Object	Wall	Slab	
	Requirements	Wall.Formwark.isComplete = True	Wall.IsPlaced = True	
	Final State	Wall.IsPlaced = True	Slab.Formwark.isComplete = True	
Unstructured Text	**Description**	"Place Wall Level 5"	"Place Slab Level 6"	
	Predecessors	"Reinforce Wall Level 5"	"Place Wall Level 5"	
			* Object Oriented Class Abstractions	

Fig. 2.3. Evolution of activity representation schemas used in the literature to formalize construction planning and scheduling knowledge.

2.3 Findings from Critical Examination of the Literature

2.3.1 Knowledge Formalization

Activity Representation

The literature employs different techniques for representing construction activities in computer-interpretable data schemas (Fig.2.3). As can be seen in Fig.2.4, 40.9% of systems use unstructured text strings to represent activity descriptions. The modeled work activities do not exhibit semantic representation, and as such, they can not explicitly offer information about which building components these activities apply to. For example, "Place walls level 5" and "Pour verticals 5th floor" are two entirely different activities. For these types of activities, these planning methods and systems do not offer any automated mechanism for "interpreting" construction activities descriptions. In other words, an activity description is nothing but an unstructured string of text. Examples are shown in [44], [45], [46], [47], and [48] as well as in older contributions such as [49], [50], [51], [52], [53], and [54].

Another 46.3% of existing methods and systems employ an object-oriented activity representation. Relevant examples in the literature include [55], [56], [57], [58], and [59] where activities are part of a hierarchy of classes that share similar attributes. Hence, work activities are dynamically

Scope Quantification	Activity Formalization[1]	Modeling Work Sequences[2]	Year	Authors	Hierarchical Planning[3]	Characterizing Work Locations[4]	Knowledge-base Population[5]	Continuous Learning[5]	Optimization Capabilities[3]	Scope[6]	Applicability[7]
Model-based	OOS	PT	**1998**	Aalami & Fischer	S		U	U		G	P
			1996	Fischer & Aalami	S		U	U		G	Sc
			1994	Fischer et al.			U	U		G	Sc
		SR	**1993**	Winstanley et al.	S	L, H	U			G	P
			1992	Winstanley & Hoshi	S	L, H	U			G	P
	OO	PT	**2015**	Liu et al.		L, H	U		S	Sp	P
			2010	Wu et al.	S		U	U		Sp	Si
			1997	Rankin et al.	S		U			G	P
			1994	Kartam		R	U			G	P
		CR	**1994**	Morad & Beliveau		H	U	U		G	Sc
			1991	Morad & Beliveau		H	U	U		G	Sc
		SR	**2012**	Hartmann et al.			U	U		G	Sc
			1991	Cherneff et al.	S	L, H	U			G	P
		P	**2013**	Chen et al.			U			G	Si
			2009	Tauscher et al.		H	U	U		G	P
	F	F	**2017**	Yeoh et al.		H	U			G	P
	I	CR	**2015**	Kim & Cho		L, H	U			G	Sc
			2014	Faghihi et al.			U		S	G	Sc
	C	P & CR	**2004**	Akbaş		H	U			G	P
	T	PT	**2019**	Naticchia et al.		H	U		S	Sp	Sc
			2018	Volk et al.		H	U		S	Sp	Sc
			2018	Naticchia et al.		H	U		S	Sp	Sc
			2016	Mohammadi et al.		L, H	U			Sp	P
			2016	Weldu		L, H	U	U		Sp	P
			2016	Haiati et al.	S	L, H	U			Sp	Sc
			2015	Park & Cai	S	L	U			G	Sc
			2014	Tauscher et al.		L, H	U	U		G	P
			2014	Wang et al.		H	U			G	Si
			2014	Morkos		R	U	U		G	P
			2012	König et al.			U	U		G	Si
			2008	Kataoka		L, H	U	U		G	P

[1] **OOS**: Object Oriented - Intersection of Semantic Entities; **OO**: Object Oriented - Description with Different Attributes; **F**: Operator, Functional Requirement or Component State; **I**: Generic "Install" applied to Components; **C**: Crews Acting on Locations; **T**: Text Description
[2] **PT**: Process Templates; **CR**: Physical Components Relationships; **AR**: Activity-based "If-Then" Rules; **SR**: Component/State-based "If-Then Rules"; **P**: Predefined Activity-On-Node or Feature Logic Dependencies; **F**: Component/State-based Function Provider/Receiver
[3] **S** : Supported
[4] **L**: Locations as a List - Levels; **H**: Hierarchical Locations - Zones; **F**: Location (Flow Perspective); **R**: Location (Resource View)
[5] **U**: User-Defined
[6] **G**: Generic; **Sp**:Project or Operation-Specific
[7] **P**: Planning and Scheduling; **Sc**: Scheduling Only; **Si**: Simulation

Table 2.1. An examination of planning and scheduling methods and systems in the literature against criteria and questions defined in section 2.2.1.

Scope Quantification	Activity Formalization[1]	Modeling Work Sequences[2]	Year	Authors	Hierarchical Planning[3]	Characterizing Work Locations[4]	Knowledge-base Population[5]	Continuous Learning[5]	Optimization Capabilities[3]	Scope[6]	Applicability[7]
Model-based	T	CR	**2013**	Kim et al.	S	L	U			G	Sc
			2007	De Vries & Harink		L	U			G	Sc
			2006	Feng & Chen		L, H	U			G	Sc
			2005	Nguyen		H	U			Sp	Sc
		CR & PT	**2019**	Wang & Azar		L	U			Sp	P
			1998	McKinney & Fischer			U	U		G	P
		AR	**2001**	Wang		H	U			G	P
		P	**2015**	Volk et al.		H	U		S	G	Sc
			2010	Mikulakova et al.		H	U	U		G	P
			2009	Firat et al.		L, H	U			G	Sc
			2008	Firat et al.		L, H	U	U		G	P
User-Defined (Manual)	OOS	PT	**2017**	Garcia-Lopez		F	U			Sp	Sc
			2016	Garcia-Lopez & Fischer		F	U			Sp	Sc
			2013	Dong et al.		H	U			G	Si
			2012	Dong		H	U			G	Si
		SR	**1989**	Darwich et al.	S	L	U			G	P
	OO	PT	**1999**	Rankin et al.	S		U	U		G	P
			1997	Sacks & Warszawski	S	L, H	U	U		G	Sc
			1995	Shaked & Warszawski	S	L, H	U	U		G	Sc
			1994	Tommelein et al.	S		U	U		G	Si
			1992	Shaked & Warszawski		L	U			Sp	P
			1988	Navinchandra et al.	S		U			G	P
		CR & PT	**1996**	Stumpf et al.	S	H	U			G	P
		AR	**1990**	Moselhi & Nicholas			U			G	Sc
			1990	De La Garza et al.	S	L	U				Sc
			1988	De La Garza	S	L	U				Sc
		SR	**1991**	Echeverry	S	R	U			Sp	P
			1990	Zozaya-Gorostiza et al.		L, H	U			G	P
			1988	Zozaya-Gorostiza et al.		L, H	U			G	P
			1987	Hendrickson et al.		L, H	U			G	P
			1986	Hendrickson		L, H	U			G	P

[1] **OOS**: Object Oriented - Intersection of Semantic Entities; **OO**: Object Oriented - Description with Different Attributes; **F**: Operator, Functional Requirement or Component State; **I**: Generic "Install" applied to Components; **C**: Crews Acting on Locations; **T**: Text Description
[2] **PT**: Process Templates; **CR**: Physical Components Relationships; **AR**: Activity-based "If-Then" Rules; **SR**: Component/State-based "If-Then Rules"; **P**: Predefined Activity-On-Node or Feature Logic Dependencies; **F**: Component/State-based Function Provider/Receiver
[3] **S** : Supported
[4] **L**: Locations as a List - Levels; **H**: Hierarchical Locations - Zones; **F**: Location (Flow Perspective); **R**: Location (Resource View)
[5] **U**: User-Defined
[6] **G**: Generic; **Sp**:Project or Operation-Specific
[7] **P**: Planning and Scheduling; **Sc**: Scheduling Only; **Si**: Simulation

Table 2.2. An examination of planning and scheduling methods and systems in the literature against criteria and questions defined in section 2.2.1 (Continued).

Scope Quantification	Activity Formalization[1]	Modeling Work Sequences[2]	Year	Authors	Hierarchical Planning[3]	Characterizing Work Locations[4]	Knowledge-base Population[5]	Continuous Learning[5]	Optimization Capabilities[3]	Scope[6]	Applicability[7]
User-Defined (Manual)	OO	SR	**1985**	McGartland & Hendrickson			U				
		AR & SR	**1990**	Benjamin et al.		L	U			G	P
		CR & SR	**1990**	Waugh			U			G	P
		P	**2007**	Tauscher et al.			U	U		G	P
			2005	Huang & Sun		H	U			Sp	Sc
			1999	Tah et al.			U	U		Sp	P
			1997	Dzeng & Tommelein	S		U	U		Sp	P
			1992	Yau	S	L	U			Sp	P
	F	F	**2013**	Chua et al.		R	U		S	G	Sc
			2006	Huhnt & Enge			U	U		G	Sc
			1991	Kartam et al.	S		U			G	P
			1989	Kartam			U			G	P
	I	CR	**1997**	Adjei-Kumit & Retik			U	U		G	P
	T	PT	**2008**	Xu & Muñoz-Avila	S		U	U		G	Sc
			2001	Muñoz-Avila et al.	S		U	U		G	P
		AR	**2004**	Dzeng & Lee			U	U		G	
		PT & AR	**1991**	Sirajuddin & Mawdesley		H	U			G	P
			1991	Mawdesley & Askew		H	U			G	P
		P	**2007**	Ryu et al.	S	L	U	U		G	P
			2006	Lee et al.			U			G	Si
			2004	Ryu & Lee			U	U		G	P
			1998	Lee et al.			U	U		G	Sc
			1994	Kähkönen		L, H	U	U		G	Sc
			1991	Al-Shawi & Jagger	S		U	U		G	P
			1990	Al-Shawi et al.			U			G	P
			1990	Kähkönen & Laurikka		H	U				Sc
		P & F	**1987**	Levitt & Kunz			U			G	Sc
			1985	Levitt & Kunz			U			G	
Undefined	Undefined	Undefined	**1991**	Soh & Lee			U				
			1990	Kano			U				
			1986	Bremdal			U			G	

[1] **OOS**: Object Oriented - Intersection of Semantic Entities; **OO**: Object Oriented - Description with Different Attributes; **F**: Operator, Functional Requirement or Component State; **I**: Generic "Install" applied to Components; **C**: Crews Acting on Locations; **T**: Text Description
[2] **PT**: Process Templates; **CR**: Physical Components Relationships; **AR**: Activity-based "If-Then" Rules; **SR**: Component/State-based "If-Then Rules"; **P**: Predefined Activity-On-Node or Feature Logic Dependencies; **F**: Component/State-based Function Provider/Receiver
[3] **S** : Supported
[4] **L**: Locations as a List - Levels; **H**: Hierarchical Locations - Zones; **F**: Location (Flow Perspective); **R**: Location (Resource View)
[5] **U**: User-Defined
[6] **G**: Generic; **Sp**:Project or Operation-Specific
[7] **P**: Planning and Scheduling; **Sc**: Scheduling Only; **Si**: Simulation

Table 2.3. An examination of planning and scheduling methods and systems in the literature against criteria and questions defined in section 2.2.1 (Continued).

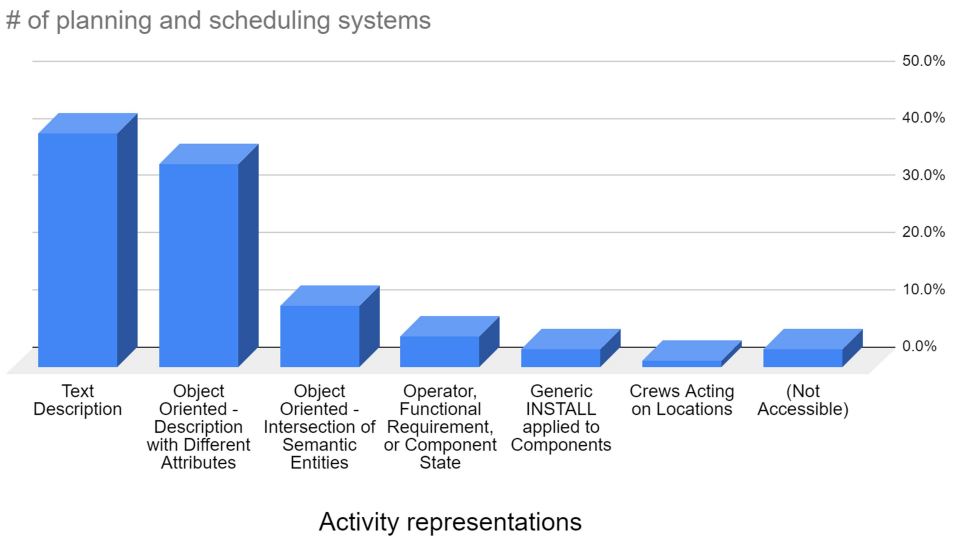


Fig. 2.4. Different representations are used to describe work activities. “Not Accessible” refers to three papers that the authors only had access to their abstracts.

linked to other project entities, such as work crews or work locations. A total of 23.3% of these relevant object-oriented or 10.8% of the total number of planning engines –as shown in Fig.2.4– use an explicit semantic representation on the constituents of an activity. Garcia-Lopez [3], Aalami and Ficher [60], and Darwiche et al. [5] offer additional structured examples of these representations.

Another 5.4% of the publications including Yeoh et al. [61], Chua et al. [62], Huhnt and Enge [63], Kartam et al. [64], and Kartam [65] follow the earlier general-purpose planning systems' definition of activities. These systems –or their so-called 'operators'– require a certain project state or preconditions to operate and produce an outcome. Here, the project state refers to a description of the work-in-progress status of different elements and conditions of the project at a specific point in time. The outcome of the 'operators' is a new modified project state. Activities in these systems are referred to as operators because they treat the planning problem as a function that requires a certain input (a set of preconditions) and produces a certain output (a new project state). For example, the task "*form slab*" requires the condition "*place wall*" of the lower level to be satisfied. Hence, the project state should include "*placing wall completed*" to enable the application of the operator "*form slab*". When that operator is applied, it modifies the project state by adding "*slab formwork complete*" (Fig.2.3). Other systems such as [66], [67], and [68] sequence building elements without using any explicit representation for activities. Akbas [69] is the only publication in this category that takes a different approach where activities are defined as movements of crews across different work locations. This new representation allows sequencing projects by maximizing the usage of work locations and improving the productivity of the work crews.

Representation of Sequencing Knowledge

Any automated planning and scheduling system should be able to reason about activity sequencing to create feasible plans. As shown in Fig.2.5, this type of reasoning is typically carried out based on the logic among work activities (shown in orange) or among building components (shown in blue). When sequencing is carried out based on the logic among activities, it is most often represented using process templates also known as fragnets (observed in 36 articles). These work templates specifically include predefined sequences of activities and many good examples are listed in [70], [44], [3], [71], and [72]. Other planning and scheduling systems (22 articles and most significantly

Representation Schema	Activities / Components	Relationship Representation
Fragnets / Process Templates	Building a wall	Reinforce → form → place concrete
Predefined Activity-on-Node Dependency	"reinforce wall level 1" & "form wall level 1"	Finish-to-Start (FS) relationship*
Activity-based If-Then Rules	"reinforce wall level 1" & "form wall level 1"	If "reinforce wall level x" is complete, then "form wall level x"
Component-based If-Then Rules	"build slab level 1" & "build walls level 1"	If all "wall" in level x are complete, then "slab" in level x
State-based If-Then Rules	"reinforce wall level 1" & "form wall level 1"	If "reinforce wall level x" in current project state, then "form wall level x"
Component Relationships	Slab & Wall	Wall *supports* slab
State-based Function Provider/Receiver	"reinforce wall level 1" & "form wall level 1"	Form(wall, level x, [Reinforce(wall, level x).isComplete()])
Component-based Function Provider/Receiver	Slab & Wall	Support(Slab, Wall)
Feature Logic Description of Constraints	"bricklaying wall" & "finishing wall"**	"bricklaying wall" is defined by a set of features such as {height=x, width=y, surface: {material = brick, height=z, width=w}} "finishing wall" prerequisites are defined as {wall, surface: {material=brick}}

*Other types of relationships include Start-to-Start, Start-to-Finish, etc..
**recreated based on Tauscher et al. 2007

Table 2.4. Evolution of relationship representation schemas used in the literature to formalize construction sequencing knowledge.

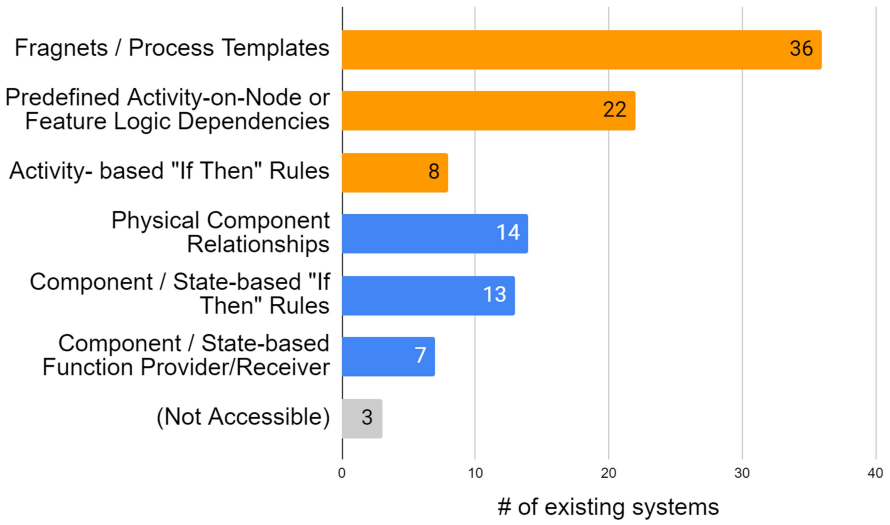


Fig. 2.5. Formal representation and modeling of scheduling logic in different automated planning and scheduling systems. "Not Accessible" refers to three papers that the authors only had access to their abstracts.

[73], [56], [74], [75], and [76]), ingest a manually defined list of activities and their dependencies as input to their engine. For example, when reading the input activities in these systems, an activity such as "*paint wall*" will be linked to an explicit list of predecessor activities such as "*build wall*" and "*prime wall*".

The more conventional expert systems such as [77], [78], [15], [2] represent sequencing knowledge and work logic using explicit conditional statements (*If-Then*), and they apply these conditional statements to common work activities. Where component-based scheduling was conducted, sequencing knowledge is modeled vis-a-vis: (1) direct relationship among different components –14 articles; e.g., [70], [66], [37], [79], and [69], (2) Conditional set of rules based on a project state defined by the presence of physical components –13 articles; e.g., [80], [81], [82], and [7], or (3) a set of component-based predicates for functional operators such as [61], [83], [63], and [64]. Table.2.4 shows examples of different relationship representation data schemas as found in the literature.

Examining techniques for modeling activity and sequencing relationships shows that these tech-

niques rely on extensive listing of activity descriptions or their dependencies in one form or another. Let's consider the following planning scenario: to plan for the scope of work involved in construction or a typical apartment building, one can list hundreds of model objects that require planning such as "columns", "walls", "slabs", "shafts", "mechanical pipes", "electrical shafts and wires", "doors", "door handles", etc. Multiple operations can also apply to each construction object –such as "frame", "reinforce", and "place" concrete elements or "build", "prime", and "paint" wall. There are also several forms of dependencies can be formed through linking different (operation, object) pairs such as "place slab level n" $\longrightarrow$ "reinforce columns level n+1". Manually and explicitly modeling each and every object, operation, and dependency is prohibitively costly and labor-intensive. This issue highlights the following knowledge gap: *There is no readily available data schema that models construction planning and sequencing knowledge and enables automated planning and scheduling without the need for explicitly and manually listing every object, activity, and required dependency.*

Hierarchical Planning, WBS Creation, and Locations

A significant body of model-based systems uses component-based planning. Because these methods and systems reason at the component-level, they produce detailed schedules that are helpful for weekly work and look-ahead plans. However, these methods and systems are considered too detailed for master scheduling needs [81]. To better organize these detailed work activities, research has focused on hierarchical planning and organizing data around work locations such that the representations can be rolled up to lower the level of detail when it is unnecessary. Only 29% of available planning methods and systems in the literature explicitly model hierarchical relationships among work activities and can detail a high-level construction activity into its low-level tasks. In other words, less than one-third of methods are capable of breaking a master schedule into detailed look-ahead plans or rolling up activities into WBS items. Examples of those systems are [84], [85], [86], [87], [38], [88], and [89] among others.

One useful approach to hierarchical planning is the location-driven grouping of activities and components. This grouping is useful because it treats work locations as a valuable or unique resource. It allows planners and project teams to better coordinate the job around the availability of

the work locations. Here, a distinction is made between methods and systems that only reason about a list of locations or levels –such as [90]– and others that reason about a hierarchical location tree defined around detailed work zones such as [91] and [92]. The level of detail in location breakdown structure can profoundly influence the functionalities of a planning method/system. For instance, if the output of an automated engine is to automatically map elements in a BIM model to schedule activities to create a 4D BIM simulation, then the zone information is crucial in identifying the most appropriate and granular mapping between building elements and schedule activities.

As shown in Fig.2.6, 39 methods and systems such as [44], [45], [61], and [71] reasoned about zones including phases, rooms, or assemblies. 33 other ones such as [70], [71], [84], and [46] associated activities and model components with a list of building levels. Among various methods and systems, the work of [11], [62], [93], and [94] explicitly modeled locations as a resource necessary for different construction activities. Building on lean construction principles, [3] and [95] also model work location as flows influencing activities. These methods and systems can analyze why certain tasks and activities are delayed by examining the necessary flows to start their execution. In addition to labor, equipment, materials, precedence, information, and external flows, *location* is also accounted for as a flow. As such, these two recent methods formally represent, measure, and track all the flow types to measure the maturity of upcoming tasks in project look-ahead plans. On the other hand, 32 other examined methods and systems –as highlighted in [67], [56], [80], and [96]– did not explicitly model location-related information. Overall, the literature is mature when it comes to modeling location information and hierarchical dependencies between model components.

2.3.2 Creation and Reuse of Knowledge Base Data

Data Entry

"**All**" existing Artificial Intelligence (AI) driven planning and scheduling methods and systems require manual formation of planning knowledge. Every activity representation, sequencing dependency, process template, or components' relationship needs to be manually defined by human experts and provided as input to the system.

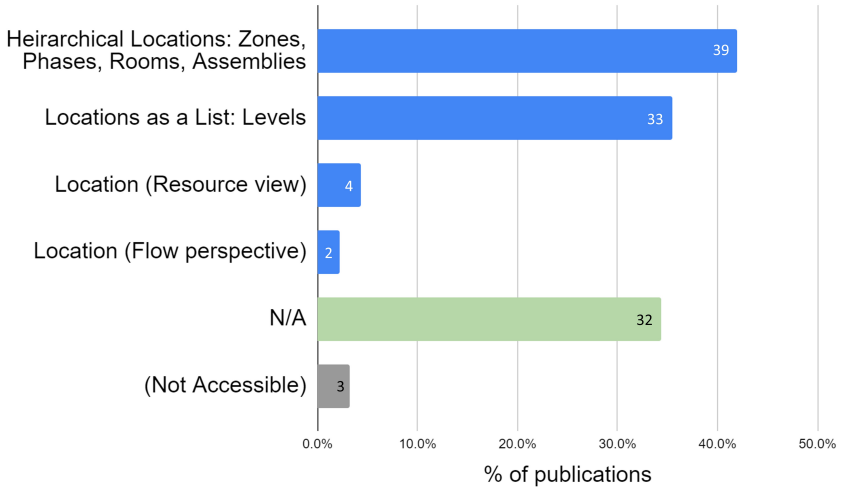


Fig. 2.6. How location is formally represented, measured, and tracked as a flow constraint. The first category includes engines that aggregate work activities together into hierarchical work locations (i.e., zones, phases, rooms, or assemblies). The second category includes engines that model and distinguish building locations in the form of a flat list (e.g., levels). The first and the second categories intersect in many automated engines. The third category represents a body of work that considers location as a resource for constraint-based scheduling purposes. The last category represents methods and systems that explicitly model how workflows across various work locations. "NA" refers to the research that did not model location information, and "Not Accessible" refers to only three papers that the authors only had access to their abstracts.

Learning Capabilities

A large portion of existing planning engines (61.3%) do not support learning from historical data associated with previous projects. Among the 35.5% that consider learning from historical data, CBR-based planners can learn by accumulating new cases in their case-based database. Examples of these methods are found in [97], [80], [74], [88], and [98]. Such methods and systems require the manual intervention of a human planner to tweak and adjust the solutions of new problems to be stored in the database for future use. Process template-based planners such as [11], [99], [87], and [100] learn by manually growing the number, diversity, and coverage of their templates. Lastly, some AI-driven planning and scheduling methods and systems can learn by manually adding activities' or components' relationships into their knowledge base [46, 68, 101, 102]. This examination highlights another knowledge gap: *There is no existing data schema or learning method that enables automated planning systems to parse through existing records and automatically learn construction knowledge without the need for extensive human input.*

Considering the manual process to create, maintain, and grow the knowledge bases and the need to explicitly list all element objects, operations, and interactions due to the rigidity of knowledge representation schemas as shown in the previous section highlights the barrier to wider adoption. Construction practitioners who are running on extremely tight schedules do not find the time to go through the efforts of manually modeling all the requirements of existing automated planning methods and system.

2.3.3 Scope and Problem Complexities

The authors examined whether existing planning solutions offer industry practitioners enough added value that would encourage them to invest the time and human resources required to manually create and manage the data required for automated planning methods and systems. As such, the literature was examined for the range and type of test projects used to validate automated planning engines. This examination is centered around better understanding of the scope and problem complexities tackled by previous research efforts. As shown in Fig.2.7, 74.2% of the efforts attempted to create a generic planning engine; i.e., a planning engine designed to assess any construction operation in any project type [103, 104, 105]. However, these planning engines are often tested only on

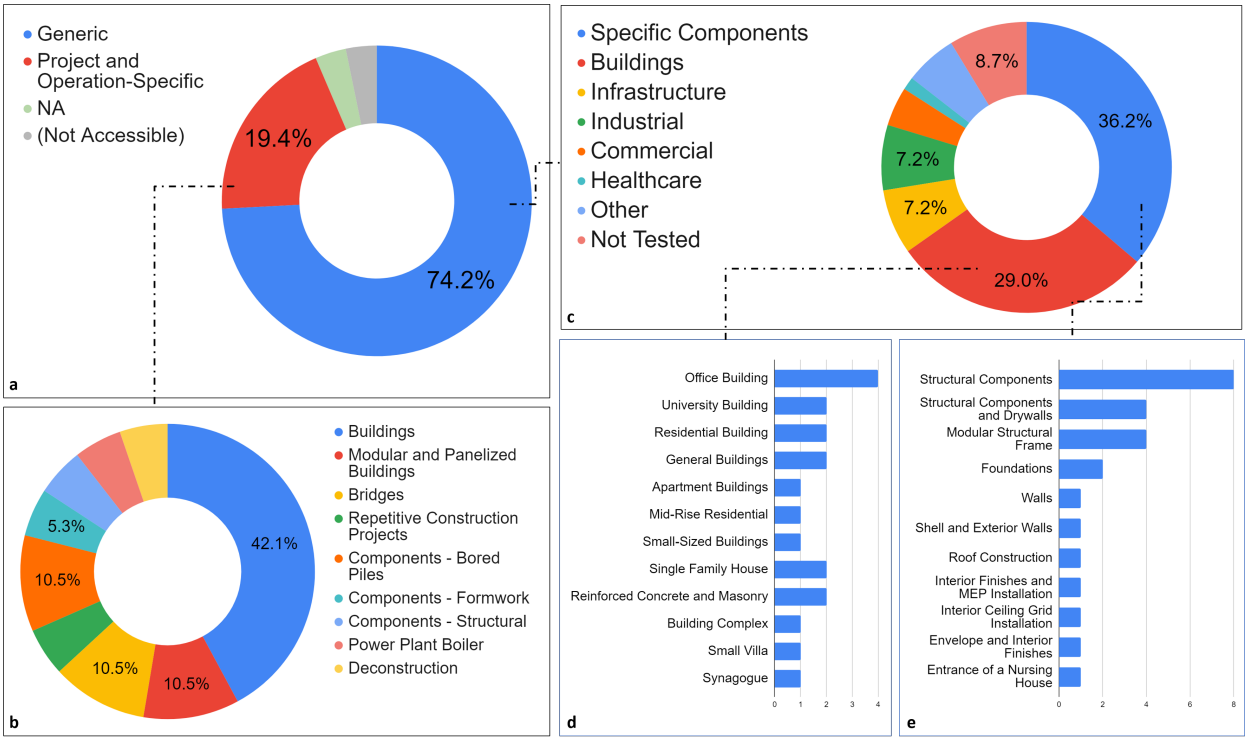


Fig. 2.7. (a) Generic planning engines are those designed to function on any construction operation. The project or operation-specific planning engines are those that are tailored to a subset of construction operations or projects. "NA" refers to the research that presented valuable contributions to the topic of automated planning –such as knowledge formalization and elicitation– without presenting a complete planning method that can be tested. "Not Accessible" refers to only three papers that the authors only had access to their abstracts; (b) The distribution of project and operation-specific engines in the literature; (c) Different types of projects are used to validate generic planning engines. "Not Tested" refers to the methods that were presented without validation in their original papers, and "Other" refers to testing cases that did not belong to a specific category such as simplified synthetic examples and very specific lab examples; (d) The distribution of project types used to validate generic planning engines; and (e) The distribution of component types used to validate generic planning engines.

one or two specific projects, a few types of construction operations at most, or a limited set of project components. For instance, 29% of planning engines were tested on few types of buildings such as commercial spaces (e.g., offices), institutional projects (e.g., universities), and residential buildings (e.g., apartments) [79]. On the other hand, 36.2% were only validated on subset types of building components, such as structural elements or drywalls [100, 106]. In addition to generic engines, 19.4% of automated planners specialize in a subset of construction operations, such as deconstruction operations or boring piles, or specific types of projects such as buildings, bridges, or repetitive construction [107, 108, 109]. Here, the authors highlight a new gap in knowledge: *The automated planning methods and systems discussed in the literature are –in most cases– not validated on real-world projects and/or throughout the entirety of a project. This simplification hides the complexities of real-world construction processes. Also from a practical standpoint, it may render the available automated planning methods systems irrelevant.*

2.3.4 Scope Quantification

As a fundamental step in creating a project schedule, automated planning and scheduling systems require an explicit mechanism for modeling the scope of the project under consideration.

Fig.2.8 shows how planning and scheduling engines have evolved over the past few decades from manual and user-driven scope quantification to model-driven methods. The earlier methods and systems such as [110], [51], [111], [112], [113], and [9] required the user to manually provide a list of activities or a list of project components to define the required scope of work. As product models such as CAD and BIM emerged in the early 90s, researchers investigated the use of geometric representation of building elements within these models to inform AI-driven engines [81, 82, 114, 115]. Throughout the 90s and 2000s, Object-Oriented CAD –such as CIFECAD [116]– started encoding additional information about project components and researchers could now leverage additional components attributes such as their types, families, and project locations [117, 118, 119, 69, 77, 120, 121, 122]. As shown in Fig.2.9, while 51.6% of previously developed planning engines required a manually defined set of activities as opposed to 45.2% that could use product models to automatically reason about the project scope, the most recent systems such as those shown in [70], [44], [61], and [46] are model-based. These systems mostly use the underlying

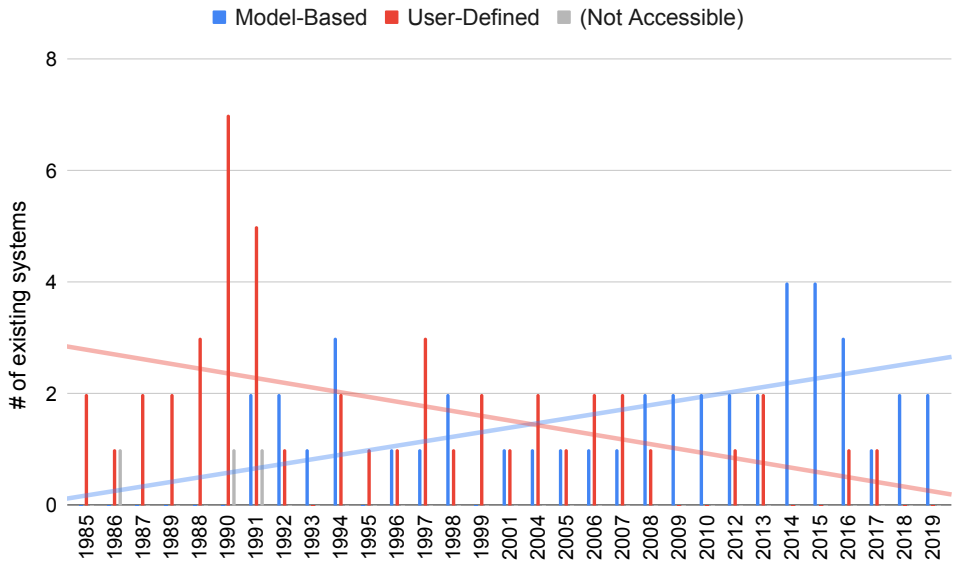


Fig. 2.8. Evolution of publications between 1985 to 2019. This graph represents how scope quantification transformed from being a user-driven process to model-based methods. The blue trend line shows the gradual birth of model-based methods, and the red trend line shows the decline of user-driven processes.

geometrical properties embedded in BIM to infer the most relevant work sequences. Other engines such as [97] leverage the encoded inter-dependencies among BIM components to model work sequences.

Knowledge-based planning and scheduling engines were mostly developed to support two use cases: (1) creating activities and sequencing the work, or (2) reasoning about the sequencing of user-defined activities or model components. 47.8% of the investigated planning and scheduling engines, such as [70], [61], [71], [46], [55], [97], [11], and [74], are well capable of creating and sequencing work activities. 45.7% of these methods and systems, such as [44], [45], [3], [84], [85], [66], [67], and [56], used their user-defined knowledge base to schedule a set of predefined activities or conduct scheduling simulations to serve different goals. For instance, [44], [3], [56], [60], [122], [123], and [124] performed scheduling operations for the purpose of analyzing flow and enabling field-driven work tracing and project controls. [6] and [99] performed simulations to assess the feasibility of multiple work execution scenarios and automatically produce look-ahead schedules. [84], [85], [125], and [126] aimed at identifying possible sequences for installing different model components. [127] deployed construction planning and sequencing knowledge for constructibility

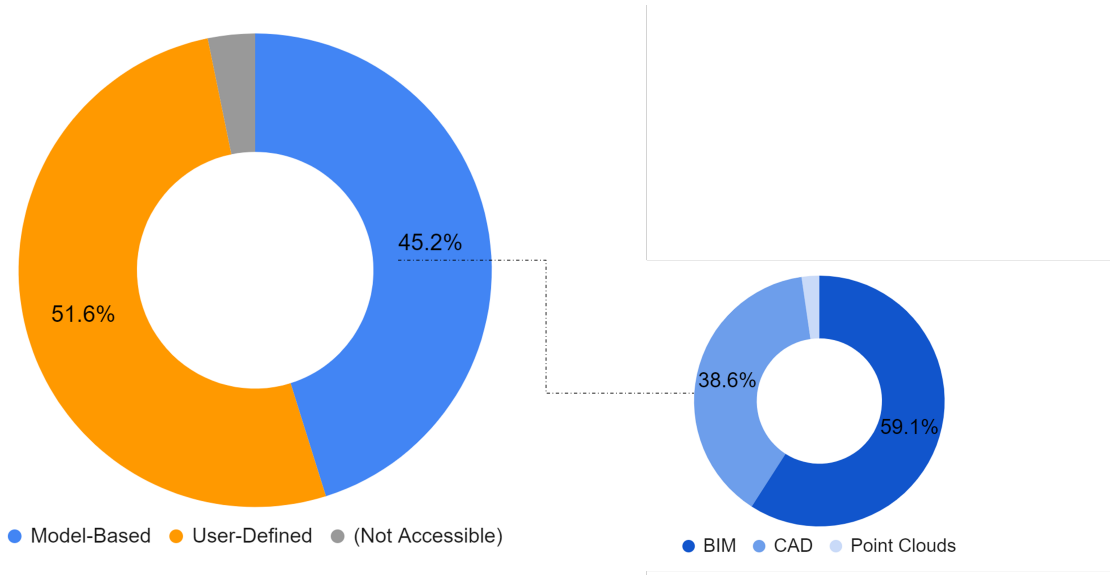


Fig. 2.9. User-driven vs. model-based methods for scope quantification.

analysis. Similarly, [21] leveraged a planning knowledge base to critique and assess the quality of existing schedules.

The findings suggest that existing planning engines are mature enough to directly reason about project scope from BIM models. They are also capable to create a project plan and schedule activities. However, a common thread across these efforts is the following: to enable these engines, there is always a need for a well-defined knowledge base that has to be either manually or automatically generated, whether or not the work is scoped from BIM, planning and scheduling are accounted for together, scheduling is the only focus, or different scenarios are simulated.

2.3.5 Schedule Optimization

Schedule optimization is hard to do when it is done manually, but computers are really good at it. Automated planning engines that perform optimization tasks offer an added value and are more appealing to practitioners. Therefore, the authors investigated the synergies between automated planning and scheduling research and optimization research.

Schedule optimization is studied extensively in the construction industry. [30], [128], [129], [37], [130], and [131] offer excellent reviews on these algorithms in terms of how single-objective and multi-objective schedule optimization problems are formulated and the wide range of their application to construction scheduling. Nevertheless, among the literature related to automated planning and scheduling methods and systems, a total of 83 out of the 93 examined (87%), do not use an explicit optimization pipeline for creating an optimal project schedule. Ant Colony Optimization (ACO) –such as the Holonic system presented by [44] and [132]–, Linear Integer Programming (IP) –such as the deconstruction system presented by [73] and [45]–, Particle Swarm Optimization [55], Genetic Algorithms [67], or a brute force search [83] are among the most popular optimization methods used to identify an optimal or Pareto optimal solution.

Here, the authors highlight a new gap of knowledge: *the line of research related to automated planning and scheduling is often decoupled from the line of research concerned with schedule optimization*. There is an opportunity in the integration of these solutions to one another to go beyond sub-optimized solutions. The current decoupled nature of these solutions has slowed the adoption of the existing automated planning solutions in real-world projects.

2.4 Discussion and Recommendations

As the extensive literature review in this paper demonstrates, available automated planning and scheduling engines vary in complexity. While some systems are capable of performing the complete planning task of creating and scheduling activities, others can only reason about sequencing dependencies and logic. Their scope ranges from handling a specific type of operation or system –such as constructing the structural frame– to solving the planning problem for any construction job and complete building projects.

The careful review of the automated planning literature revealed multiple gaps in knowledge: First, answering research question #1 revealed that the data schemas used to represent and store construction knowledge in existing planning systems are rigid [132, 133, 46]. These representations necessitate an exhaustive listing and modeling of all construction operations, model objects, and their interactions. Second, answering research question #2 showed that the creation and maintenance of knowledge bases is done entirely manually [134, 11]. Given the large amount of required construction templates, the manual task of creating them is extremely tedious and in some cases infeasible. Third, there are no machine learning methods that enable construction knowledge bases to be built automatically from existing data, while all the required information is already available through existing project schedules and weekly work plans. This is also due to the rigidity of the work templates as discussed earlier. As an answer to research question #3, while there are methods that enable capturing current planning cases to be reused in the future, these methods are not fully automated, and they still require manual input from users [80, 74]. Fourth, the validation of existing automated planning systems has been limited. These systems, in most cases, have not been tested on real-life construction projects [100, 107]. This oversimplification of the planning task renders the developed systems less appealing to industry practitioners. For instance, there is a lack of an intelligent planning and scheduling system that can simultaneously account for mechanical, electrical, plumbing, fire protection, and finishing works while also planning and scheduling basic structural and interior drywall operations. These finding highlight the answer of research question #4. Fifth, as for research question #5, as shown in Fig.2.8 and Fig.2.9, and as discussed in the Scope Quantification section, the findings of this paper suggest that existing planning and scheduling engines are mature enough to reason about project scope directly from BIM models

[135, 134, 117]. Sixth, regarding research question #6, while existing methods can automatically plan and schedule tasks, construction optimization research is decoupled from planning automation research. Very few automated planning engines leverage any optimization capabilities [44, 73]. Since optimization is hard to do manually by a human planner, offering optimization capabilities with automated planning engines offers additional value to industry practitioners. The decoupling in these two lines of research is creating an opportunity cost that can be leveraged in the future. These knowledge gaps can be addressed with:

- (1) new formalization methods and representations for construction planning knowledge that (a) is flexible and (b) enables automatic creation and maintenance of knowledge bases;
- (2) more rigorous testing on real-life projects in order to capture the actual complexities of the planning task;
- and (3) systems that tightly integrate automated construction planning and schedule optimization.

In regards to bullet point 1, the advancements of natural language processing, machine learning, and data mining techniques as well as the availability of large amount of schedule data and project records offer a golden opportunity. Future research should focus on the automated learning of construction planning and scheduling knowledge from existing data. The authors took early steps in this direction by developing a deep learning-based system, namely Part-of-Activity (POA) tagging, that automatically reasons about the constituents of construction activities [136]. They then integrated POA tagging within a new formalization and method to mine dynamic construction work templates from existing schedules [137]. Similar efforts were invested by [138] who used traditional syntactic parsing techniques to extract an ontology of construction tasks from existing data automatically.

In regards to bullet point 2, the authors believe the creation of large database through learning from existing records will automatically enable more rigorous testing on real-life projects. Once all the required construction templates are dynamically and automatically learned from existing records, that already reflect the complexities of real-life scenarios, then engineering and testing automated planning systems on real-world projects will be possible.

The decoupling between optimization and automated planning –bullet point 3– can be also addressed through formulating optimization problems that solve for automated planning and schedule sequencing simultaneously.

Lastly, the authors suggest that the synergies between automated planning engines and lean construction theories need to be more carefully investigated. This research explored automated planning systems' ability to perform hierarchical planning and showed that these systems can support creating plans at different levels of granularity. This exploration was particularly targeted at understanding the ability of automated planning systems to support lean workflows where the project plan is updated and detailed continuously at different levels of granularity throughout the project life-cycle. Similarly, automated planning engines that "understand" construction planning at the master schedule level as well as the look-ahead and Weekly Work Plans levels can be used to help practitioners align and maintain the alignment between short and long term plans. The new generation of automated planning systems can play a supporting role for existing lean workflows where the knowledge of human planners is augmented by artificial intelligence and its immediate access to years of previous experiences.

In terms of research limitations, this manuscript did not fully explore the synergies between lean construction and automated planning systems. What is being offered on this topic are observations that other researchers can build upon. Similarly, this research did not explore the different techniques of schedule optimization that can be implemented on top of automated planning systems. It only highlights an opportunity that future research can leverage and expand.

2.5 Conclusions

This paper provides a critical literature review on the most significant body of knowledge related to automated construction planning systems. The authors examined more than 1,565 articles and synthesized results from 93 to highlight knowledge gaps that possibly hinder a wider adoption of automated construction planning solutions. This examination revealed three insights regarding why existing planning and scheduling systems cannot scale in complexity to handle data from real-world projects. The first insight is that methods for modeling and representing activities and dependencies

do not enable automatically building knowledge bases using data from existing projects. Current planning data schemas and representations in the literature are rigid and do not provide semantic representation to generalize knowledge from existing projects to new projects. The second insight is that existing planning systems are not validated on complex real-life projects. They are often tested on simplified (or hand-made) planning problems where only a subset of building systems or operations is considered. The third insight is that there is a decoupling between optimization and automated planning research that causes automated planning systems to be less appealing to industry practitioners. Without an automated mechanism to generate these sequencing templates, adopting and adapting these techniques for repetitive and non-repetitive projects will remain a major challenge. Given the advancements in machine learning techniques, the authors offer a path forward towards wider adoption via automatic generation of these work templates from previous project schedules while carefully considering the granularity of project schedules, and how they can roll up and down from look-ahead plans to master-schedules.

2.6 Data Availability Statement

All data, models, and code generated or used during the study appear in the submitted article.

CHAPTER 3

MODELING DYNAMIC CONSTRUCTION WORK TEMPLATE FROM EXISTING SCHEDULING RECORDS VIA SEQUENTIAL MACHINE LEARNING [2]

[2]This chapter is published in *Advanced Engineering Informatics* [139]:
"Fouad Amer and Mani Golparvar-Fard, "Modeling dynamic construction work template from existing scheduling records via sequential machine learning", *Advanced Engineering Informatics, vol. 47*, Jan 2021, https://doi.org/10.1016/j.aei.2020.101198"

3.1 Introduction

Over the past three decades, a large body of research has focused on automated planning and scheduling for construction. Early efforts focused on using expert systems to automate the process of planning; i.e., creating and sequencing schedule activities given a project scope [9], and scheduling; i.e., time stamping different activities considering best practices and project-specific constraints [52]. Recent efforts have focused on automated construction schedule generation; e.g., [70, 135, 46, 71, 134], computer simulation; e.g., [140, 141], genetic algorithms for schedule optimization; e.g., [142, 143, 29, 144, 145], or an integrated system comprising of two or all of these methods; e.g., [44, 146, 147]. Several recent publications such as [35, 148, 149] offer a comprehensive reviews of these techniques.

While substantial research is conducted, yet the feasibility of scaling these methods to real-world projects is not fully examined. The complexities of actual construction projects on one hand, and the nuances between different construction approaches, on the other, have rendered the manual creation of explicit knowledge representations extremely tedious and, in many cases, entirely infeasible. Hence, most –if not all– construction projects still rely on a single or a small group of experienced practitioners to manually perform their planning and scheduling activities. Current project-specific approaches do not benefit from years of experience accumulated by other planners

in similar situations and, in turn, can not guarantee the optimality of generated plans. Given the availability of large-scale historical data, machine learning poses a potential solution to tackle the scalability issue associated with capturing and sharing planning and scheduling knowledge. As such, our hypothesis is *to explore if planning and scheduling knowledge and work templates can be modeled using machine learning, with minimal human input*.

Activities –the most common basic scheduling units in construction– and their connections form a heterogeneous directed graph; where each node in the graph represents an activity and each edge represents a precedence connection. Nevertheless, the information that describes each schedule activity (i.e., *activity descriptions*) is unstructured text. Hence, a machine learning technique for modeling planning and sequencing knowledge using previous schedule data, should be able to handle the unstructured nature of the data presented in *schedule activity descriptions* and the structured data presented by the *logical dependencies*.

Learning a model from a wide range of schedules developed by different planners requires decoding semantics in *activity descriptions*. For example, "place concrete level 2" and "concrete placement second floor" refer to the same activity. Similarly, "F/R/P columns - lvl 5" and "concrete columns 5th floor" also refer to the same activity where "F/R/P" is a specific term used in construction scheduling that compresses "framing", "reinforcing", and "placing" into one word, and "lvl" is an informal representation of "level". Hence, formalizing semantics for activity representations that can be generalized to different types of projects is the first step towards the mining of sequencing knowledge and the overall learning task.

In addition to formalizing activity representations, modeling sequencing logic and representing precedence dependencies also requires structured learning. A sequential learning model can learn the distribution of activity successors over a sequence of predecessors allowing for the prediction and inference of the most likely successor activities. And, a similar model can be trained to learn the distribution of predecessors over a sequence of successors allowing for the identification of predecessor activities necessary to complete the work leading to a single or a sequence of future activities.

To address these needs, this paper presents a new machine learning driven method that models

construction planning and sequencing knowledge. We build on preliminary results of our earlier work [137] and present a generative sequential Long Short-Term Memory Recurrent Neural Networks (LSTM-RNN) that learns company-specific and project-type specific precedence dependencies, using schedule data from previous projects. Provided with a sequence of schedule activities as an input, the learned model can predict and infer activities that are most likely to appear next. At any point in time during the project execution, our learned model can generate a set of suggested activities for the remaining scope of work. The model can also be used for look-ahead planning purposes. Specifically, for any given activity, our model can predict what activities need to take place as its predecessors by considering the logical constraints associated with the remaining scope of work. Hence, given a list of upcoming activities, our model can predict what predecessors should have been completed by today's date to "make ready" the future activities. We present a comprehensive set of experiments to validate the presented machine learning method using schedules obtained from real-world construction projects.

The following section provides an overview of the literature. Next, we present our method along with the data and experimental setup used for training and testing. Lastly, the results, benefits, and limitations are discussed in detail.

3.2 Related Work

Automatically learning construction planning and sequencing knowledge from previous scheduling data, requires a formal representation of construction activities and their logical dependencies. It also requires learning models from semi-structured data in *master schedules*, *look-aheads*, and *weekly work plans*.

The literature is rich when it comes to methods for formalizing schedule activities. Recent methods [44, 45, 46] and early Knowledge-based Expert Systems (KBS) [9, 113, 150] only use simple text statements to describe construction activities, while [61, 62, 63] define activities as "operators" or "functions" that act on a specific state of a project (i.e., a set of preconditions) to modify them. Akbas [69] is the first to formally define activities as a task conducted by a field crew at specific project location. Using such a definition, their work optimizes crew movements and

overall project productivity. Liu et al.[55] and Chen et al.[56] among many others, leveraged object-oriented schemes to explicitly model the connections between activities and resources, locations, and other entities that influence project execution. Similarly, Darwiche et al. [5] are among the earliest examples of introducing semantic object-oriented representations for modeling *activity description*. In this work, *activity descriptions* are defined with {Object, Action, Resources} (OAR) tuples. Fischer, Aalami and others in [127] and [60] expanded the OAR representations into the {Constituents [Objects], Actions, Resources, Sequencing Constraints} or CARS schema. The new schema was defined as *Construction Method Model Templates* (CMMTs). Recent efforts such as Garcia-Lopez [151] build on the concept of *flow* from lean construction literature [152] to explicitly model the influence of labor, materials, spaces, equipment, precedence, information, and external factors on activities represented with CAR tuples for project tracking purposes. Using hard-coded expert rules and sequencing heuristics, [115], [153], [6] and [11] introduce schemas for project planning and control needs. Their approach challenges traditional representations of execution plans that primarily rely on activities linked by precedence relationships to express work that can be done at different states of a project. Instead, they categorize the 'STATUS' of activities with TODO, CANDO, DOING, ENDED, and DONE to track the performance of each activity, bring more clarity to a project's 'STATUS' and improve reliability of look-ahead scheduling. However, as shown in the following sections, all previous efforts relied on an explicit hard-coded representation of activity descriptions that limits their flexibility in handling unseen scenarios and their ability to learn from existing records.

In addition to formalizing the representation of *activity descriptions*, scheduling requires formalization of sequencing dependencies. To that end, Echeverry et al. [7] categorized sequencing constraints into physical relationships, path interference, trade interactions, and code regulations and defined flexibility and time dependency attributes. Similarly, Koo et al. [10] introduced "role" and "status" attributes for scheduling constraints and leveraged these attributes to enable quick re-scheduling of the look-ahead plans. Dong et al. [6] presented sequencing knowledge using "fragnets" or predefined sequences of activities that describe a process. Their work introduced "zone" and "blocking" constraints specifically for the closeout phase of construction projects. Sequencing constraints were also leveraged by Han et al. [8] for the purpose of reasoning about state

of work in progress where as-built observations are incomplete or are constrained with limited visibility. These "sequencing constraints" are enforced and applied to planning and scheduling problems in the form of (a) process templates [70, 151, 71], (b) If-Then statements [77, 78, 15], or (c) state-based conditional requirements [61, 62, 80, 81] which are all hard coded in the planning engines' internal logic. Such representations reduce the flexibility in handling new planning scenarios, and as such, they may not scale well to all scenarios.

To address the need for automated learning from novel observations and growing knowledge bases, Case-based Reasoning (CBR) methods were introduced. In these methods, planning problems are described with specific project features and are recorded in the knowledge base along with their solutions as individual cases [134, 80, 74, 88]. When planning for a new project, a CBR system first identifies new project features and retrieves the closest case from the knowledge base following a set of predefined similarity criteria. Next, considering project-specific features, the retrieved solution is adapted to address the new problem. As shown by several research projects such as [74], designing a robust adaptation process is difficult, and as such, it is often excluded from the design of CBR systems. Hence, practitioners would need to manually perform such tasks to apply the knowledge base to their project-specific requirements.

As an alternative learning method, Zhao et al. [138] recently used traditional syntactic parsing to learn a construction scheduling ontology from existing schedules. Sigalov and Konig [154] introduced a new method for mining process patterns from construction schedules. Their method relies on schedules developed by human planners within the method's knowledge base following specific rules and restrictions on *activity descriptions* and relationships. As such, automatically learning from historical construction schedule data was not in their scope of work. Learning from existing scheduling records and using a data-driven approach to model planning and sequencing knowledge necessitates activity representations to generalize over semi-structured schedule data extracted from multiple projects. This data mining should be designed and implemented such that (a) the differences between different projects, (b) the repeated logic in different locations, and (c) the differences in natural language expressions can be handled systematically.

Automated planning and scheduling has been a subject of research for many years; Nevertheless, all prior methods rely on manually structured knowledge bases. Even recent automated schedul-

ing and optimization solutions in the market– e.g., ALICE [155]– rely on manually structured work templates or so called *recipes*. The reliance on explicit digital process templates, manually structured by planning experts, challenges the scalablity of these methods to real-world projects.

The complexities of construction projects are ever-growing, and manually modeling the interactions between different systems and components is not a trivial task. To this end, we aim at solving scalability and complexity problems of manually creating work templates using machine learning. We specifically introduce new sequential machine learning models that are trained in an unsupervised fashion using available schedule data. Our models implicitly learn sequencing dependencies by fitting probability distributions over all possible –previously seen– activities conditioned by their precedence context. Given (a) company-specific practices, (b) the sensitivity of information sharing between different organizations, and (c) the hypothesis that planning and scheduling knowledge is considered a company asset –without the loss of generality– we tailor our method to learn from company-specific historical records. The following sections describe the proposed method in details, where a mathematical formulation of the problem and experimental results on data obtained from 32 schedules and more than 78,000 activities are presented.

3.3 Method

We present a new method to implicitly learn planning and sequencing knowledge from historical scheduling data and store it in generative models called Dynamic Process Templates (DPTs). We also present a new formalization of construction activities and sequencing dependencies designed to generalize across multiple projects.

3.3.1 Formalization of Activities

The scheduling data for a project is most commonly represented in form of a semi-structured Activity on Node (AON) network. AON networks are semi-structured because the representation of dependencies between different activities is structured in a directed graph (see Figure 3.1c), while the representation of each individual activity is in the form of unstructured text (see Figure 3.1 a

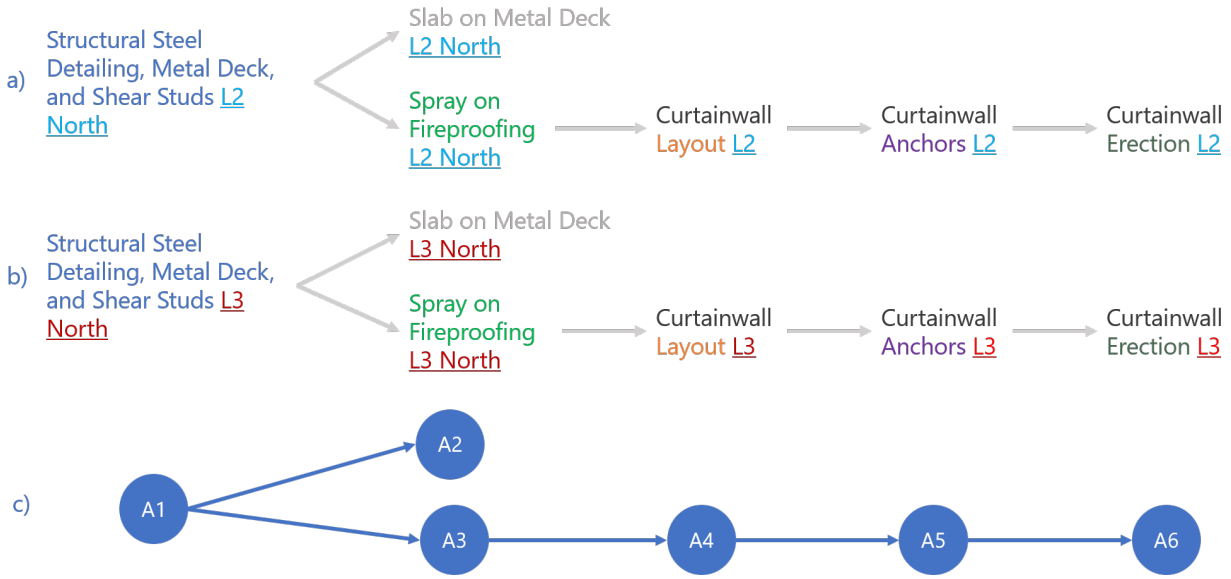


Fig. 3.1. Examples from real-world schedule activity sequences which demonstrate the semi-structured nature of scheduling data and the difficulty in generalizing activity constituents for a formal representation.

and b). The absence of a structured activity representation makes the automatic identification of *identical* activities –activities that refer to the exact same action and object while written differently– and *synonymous* activities –activities that have the same semantic meaning– non-trivial. Figure 3.1 (a) and (b) offer examples from real-world project schedules where the same sequencing logic is applied to the same activity type, while the activities are scheduled for execution at different project-specific locations.

Table 3.1 demonstrates the complexities of learning from multiple sources of historical scheduling data. In this table, several examples are shown on how schedule *activity descriptions* can be similar to one another. A machine learning based model should handle various *similarities* that are shown in this table. To address this issue, we define similarity type as an attribute with *identical* or *synonymous* as values to specifically account for the two scheduling scenarios that were discussed above. To model these similarity type attributes while accounting for their variability, the following steps are conducted:

- Each *activity description* is pre-processed and lemmatized, automatically. Pre-processing includes lowercasing, and removing numbers, punctuation and unnecessary characters. Lemmatization includes replacing each word with its *lemma* or its most basic dictionary

Table 3.1. Different forms of similarities across schedule activities.

Case	Activity A	Activity B	Similarity Type	Description
1	“Curtainwalls layout"	“curtainwall layout"	**Identical**	Exactly same *activity description*
2	“cw layout lv 2"	“cw layout lv 3"	**Identical**	Same *activity description*; location different
3	“owner A move in"	“owner B move in"	**Identical**	Same *activity description*; responsible party different
4	“cut soldier piles"	“cut soldier piles at elev. 3 ft"	**Identical**	Same *activity description*; additional project-specific info
5	“cwall layout"	“curtainwall lay out"	**Identical**	Same *activity description*, expressed using different abbreviations
6	“f/r/p slab"	“frame slab"	**Synonymous**	Different degrees of action details
7	“place columns level 2"	“concrete level 2"	**Synonymous**	Different degrees of object (scope) details
8	“pour columns level 2"	“place concrete cols second floor"	**Synonymous *or* Identical**	Similar semantic meaning but no/few common tokens
9	“f/r/p walls"	“concrete placement level 5"	**Synonymous *or* Identical**	Any combination of the above cases

form. This process addresses similarity type #1 shown in Table 3.1.

- To handle similarity type #2, #3, and #4, project-specific information or information related to " work locations" and "responsible parties" are removed from activity descriptions. As such, each activity is strictly defined by its *Action* and *Object* constituents. To identify *Action* and *Object* activity constituents, we deploy the Part-of-Activity (POA) Tagging method – previously developed in [136]– and identify the most relevant constituent associated with each work in the activity descriptions. Limiting an activity description to its *Action* and *Object* constituents is aimed at decreasing the confusion caused by project-specific and location-specific attributes. By doing so, we enable matching activities across different projects and across different locations in the same project. A high-level overview of the POA tagging method is discussed in the following section.

- We use activity embeddings to solve similarity Cases #5 to #9. An activity embedding is a vector representation aggregated over word embeddings of the *Action* and *Object* constituents of the *activity description*.

In the following, we present the pre-processing and lemmatization steps and discuss why learning scheduling-specific word embeddings is needed. We also introduce our novel learning scheme which is tailored to construction scheduling as it specifically accounts for activity dependencies and scheduling logic.

Word Embeddings for Schedule Activity Descriptions

Word embeddings are low-dimensional vector representations of real numbers that can capture distributional similarities. The similarity among these vectors can then be quantified using distance measurements. Word embeddings can be learned in an unsupervised fashion using techniques such as Word2Vec [156], GloVe [157], and FastText [158]. These methods leverage the surroundings of a word to predict that word itself (Continuous Bag of Words model) or the given word to predict its surroundings (Skip-gram model). These techniques have been used in the construction research

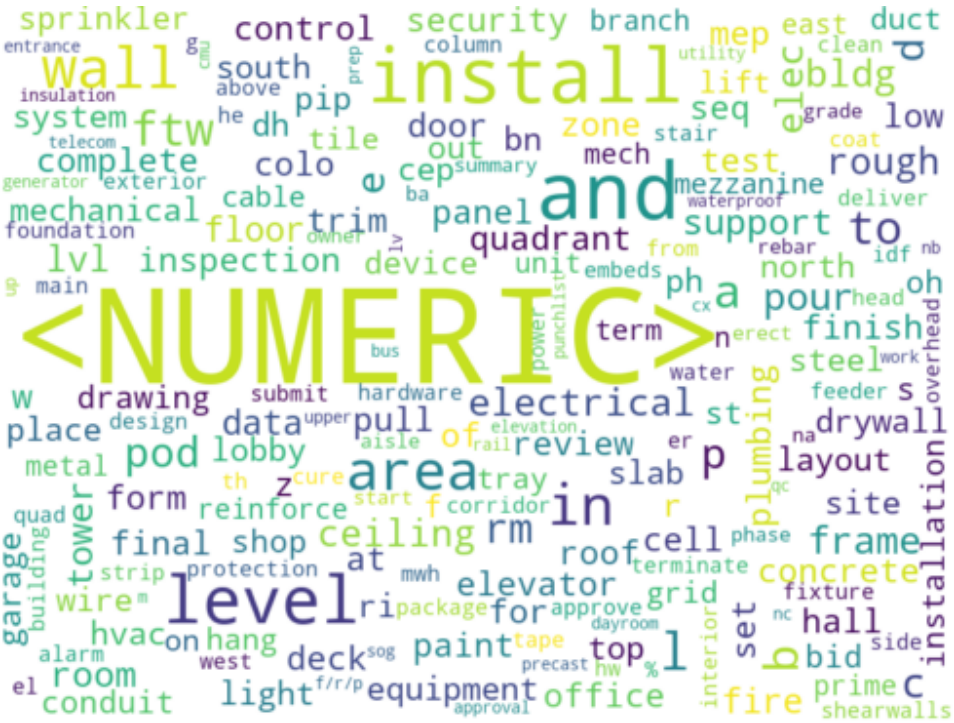


Fig. 3.2. Most common construction scheduling tokens scaled by their frequency of occurrence in historical scheduling data. These examples show (1) scheduling-specific terms {“f/r/p”, “rough”, etc.}, informal English expressions and abbreviations {“cmu”, “he”, “dh”, “rm”, etc.}, and multiple references to express the same semantic meaning {“level”, “lvl”, “lv”, etc.}

community. For example, Tixier et al. [159] trained word embeddings on a corpus of open-access web articles related specifically to construction for injury report classification purposes. As shown in Figure 3.2, schedules from real construction projects use scheduling-specific terms {“f/r/p”, “rough”, etc.}, informal English expressions and abbreviations {“cmu”, “he”, “dh”, “rm”, etc.}, and multiple references to express the same semantic meaning {“level”, “lvl”, “lv”, etc.}. Therefore, similar to [159], we hypothesize that scheduling terminology –specially the frequent use of informal language and improper grammar– requires training of construction scheduling-specific word embeddings to recognize similarities between abbreviations such as “cw” vs.“curtainwalls” or “curtain walls” and semantics such as “place concrete” vs. “pour concrete”.

To account for domain-specific characteristics, we train word embeddings in form of 320-dimensional vector representations on *activity descriptions* extracted from real-world historical project schedule data. To build the training data, each *activity description* is automatically pre-processed, tokenized, and lemmatized. The pre-processing phase removes punctuation and special characters such as {()?!*$#}, sets all the characters to lower case, and replaces numbers by the special token "NUMERIC". The tokenization process splits each *activity description* into a list of tokens. During lemmatization, each token is replaced by its *lemma*. For example, "placing" is changed to "place".

Next, word embeddings are trained using FastText [158]. To do so, we built a document by augmenting the context of *activity descriptions* as follows: each sentence in the document is composed of an *activity description* preceded by the special token "PRED" and a predecessor *activity description* and succeeded by the special token "SUCC" and a successor *activity description*. "PRED" and "SUCC" tokens maintain the precedence dependencies between different activities in the augmented sentence. This augmentation tailors the training towards construction scheduling by allowing words appearing in predecessor activities such as "anchors" and "layout" (as shown in Figure 3.1) to have similar vector representations; hence training on individual *activity descriptions* is avoided. FastText embeddings are chosen because they learn sub-word representations which can better handle abbreviations and can embed tokens that were not seen during the training process. Handling unseen tokens without FastText embeddings would otherwise necessitate the use of a special token –such as "UNKN"– to represent all unknown words.

The results of this phase –the learned word embeddings– are 320D vector representations for every single word in the scheduling vocabulary. These word embeddings are used to train a POA tagging model as discussed in the next section.

Part-of-Activity Tagging

To allow our DPTs to learn the significance and usefulness of all tokens in an *activity description* from a construction point of view, we decode the constructional functionality of each token using the Part-of-Activity (POA) tagging method of [136]. Using this method, *activity descriptions*

such as “f/r/p slab 2nd floor zone A” will be split into the following tokens [“f/r/p”, “slab”, “2nd”, “floor”, “zone”, “A”]. The *activity description* will then be tagged as [*Action, Object, Location, Location, Location, Location*]. Our set of constructional functionalities is defined as {*Action, Location, Object, Responsible Party, Extra Information, Preposition, Conjunction*}. This set includes the activity constituents: {*Action, Location, Object, Responsible Party*} or *{ALOR}*. The remaining constructional functionalities: {*Extra Information, Preposition, Conjunction*} are needed to complete the tagging process. Extra Information is used to train the model the type of information that is not essential to learning sequencing knowledge. For example, the length of piles to be cut will be ignored. Similarly, Conjunction and Preposition are needed to allow the model to identify terms such as “in” or “out”, as sometimes these terms may appear in *Action* or *Object* constituents. For example, consider “out” in “trim out” or “in” in “rough in” which can be tagged as *Action* as opposed to *Preposition*. The set of constituents presented here is not exhaustive, and it can be fine-tuned based on the required downstream tasks. For example, *Action* could be further split into *Building Action*, *Procurement Action*, *Inspection Action*, etc. Nevertheless, in this paper, we adapt the *{ALOR}* set of representations tailored towards capturing the information that is more likely to be present in an *activity description* and that would serve the scheduler’s purpose. The POA Tagger is composed of 4 bidirectional LSTM layers of size 320 (same as the embedding size), a fully connected layer of size 320 x 7 and a softmax layer to create a probability distribution over all possible tags. The model is trained for 50 epochs using ADAM [160] for optimization, a batch size of 100, a learning rate of 0.0005, and a dropout probability of 0.4 for the LSTM layers. In this paper, we briefly described the POA tagging method to offer a complete view of the proposed DPT learning method. For more details on POA model architecture or validation, interested readers are encouraged to look into [136]. In the next section, we describe how word embeddings and activity constituents are aggregated to compute representative and generalizable activity embeddings.

Activity Embeddings

The POA tagging identifies the constructional functionality of all tokens in every *activity description*. After the POA tagging, these tokens that are classified into *Action* or *Object* are maintained to represent the activity, while all other tokens are discarded. For each *activity description*, a

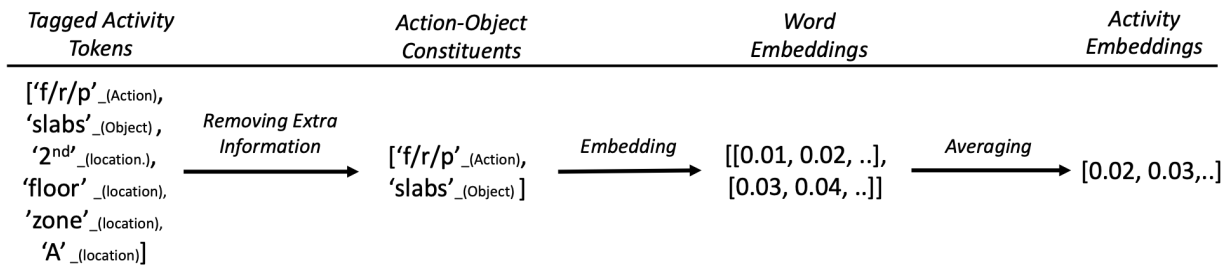


Fig. 3.3. An example that shows how activity embeddings are created using tagged *activity descriptions*.

generalizable embedding vector is computed as the average of embeddings of all tokens tagged as *Action* or *Object*. Figure 3.3 shows this procedure. Mathematically, the vector representation of a construction activity, a_{vec}, is defined based on the *Action* and *Object* constituents as follows:

$$a_{vec} = \frac{1}{|T_e|} \sum_{i}^{|T_e|} T_{e_i} \quad \text{for} \quad T_{ei} \subset a_{Act,Obj} \tag{3.1}$$

where T_{e_i} is the k_E-dimensional word embedding of token i in *activity description* a'as s set of *Action* and *Object* constituents $a_{Act,Obj}$. k_E is the size of each word embedding vector representation and is empirically set to 320.

3.3.2 Formalization of Schedule Logic and Activity Dependencies

Unlike the unstructured *activity descriptions* in project schedules, the representation of dependencies between different activities is structured in form of a directed graph. Assuming that each activity is a node, there exists a directed edge connecting each predecessor node to each of its successor nodes. To train sequential machine learning models on existing project schedules, we transform schedule graphs into linear sequences of activities with a fixed length.

This approach faces a computational barrier: the complexity of listing all sequences from an input graph is exponential with respect to the sequence length: $O({K_{A_{depth}}}^{K_S})$ where $K_{A_{depth}}$ is the number of successors/predecessors at a certain network depth (branching factor) and K_S is the length of the sequence. The Data and Experimental Setup section shows empirical evidence on the exponential increase in the number of sequences and computational complexity relative to the

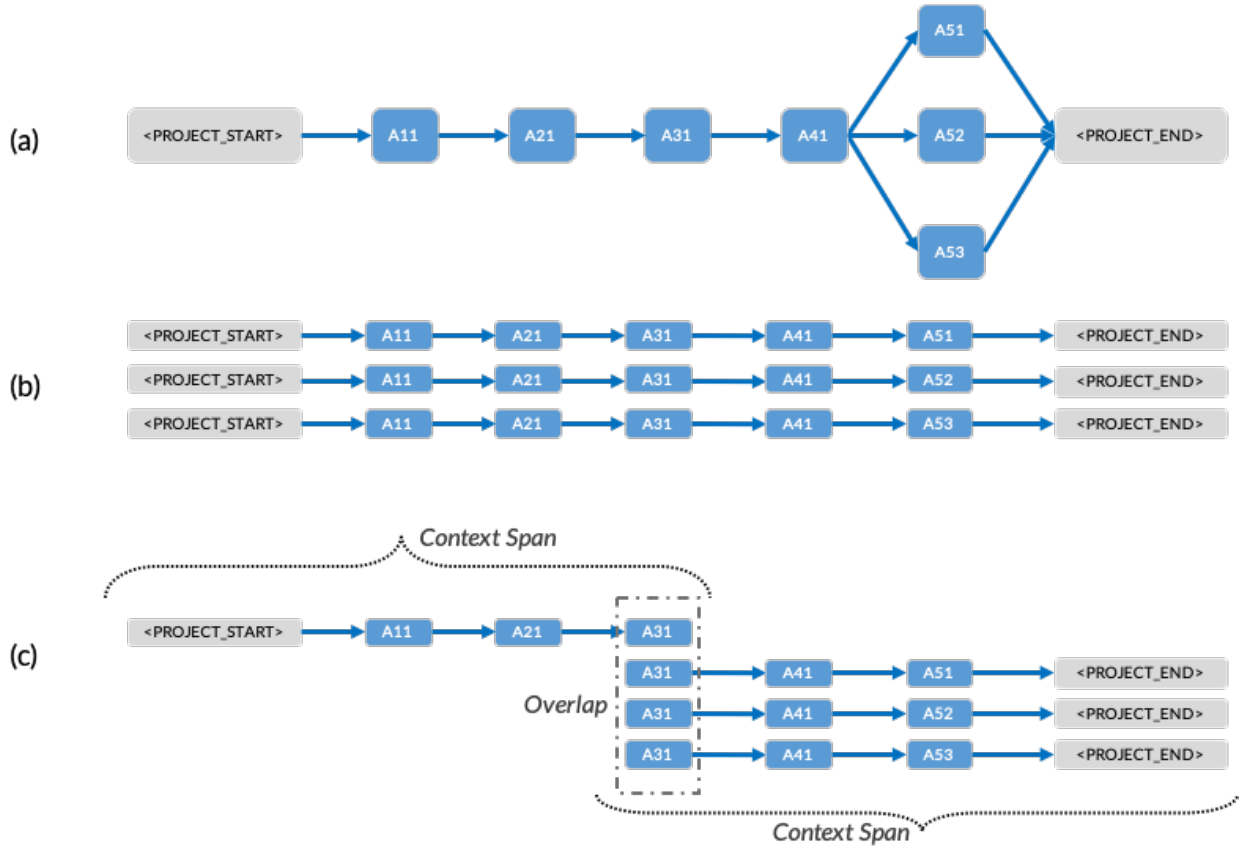


Fig. 3.4. Example of linear sequences extracted from schedule graphs. (a) shows an example schedule network. (b) shows the resulting linear sequences if the full sequence length is used. (c) shows the resulting truncated linear sequences for an overlap of one and sequence length of four. The overlap allows the extraction of shorter sequences while preserving available planning information at its entirety.

increase in sequence length.

To solve this computational barrier, we specify a fixed context span/sequence length. Figure 3.4a shows a toy example of a schedule graph, and Figure 3.4b shows all the extracted sequences if the entire sequence length as found in the graph is used. Figure 3.4c shows the extracted sequences using a maximum truncated sequence length of four and an overlap of one. When adding only one activity at a position close to the start of the graph (see Figure 3.5), the number of extracted full length sequences doubled while the number of truncated sequences only increased by one.

Modeling logic and activity dependencies with overlapping sequences allows our model to learn dependencies that extend beyond immediate predecessor/successor relationships. Specifically, it

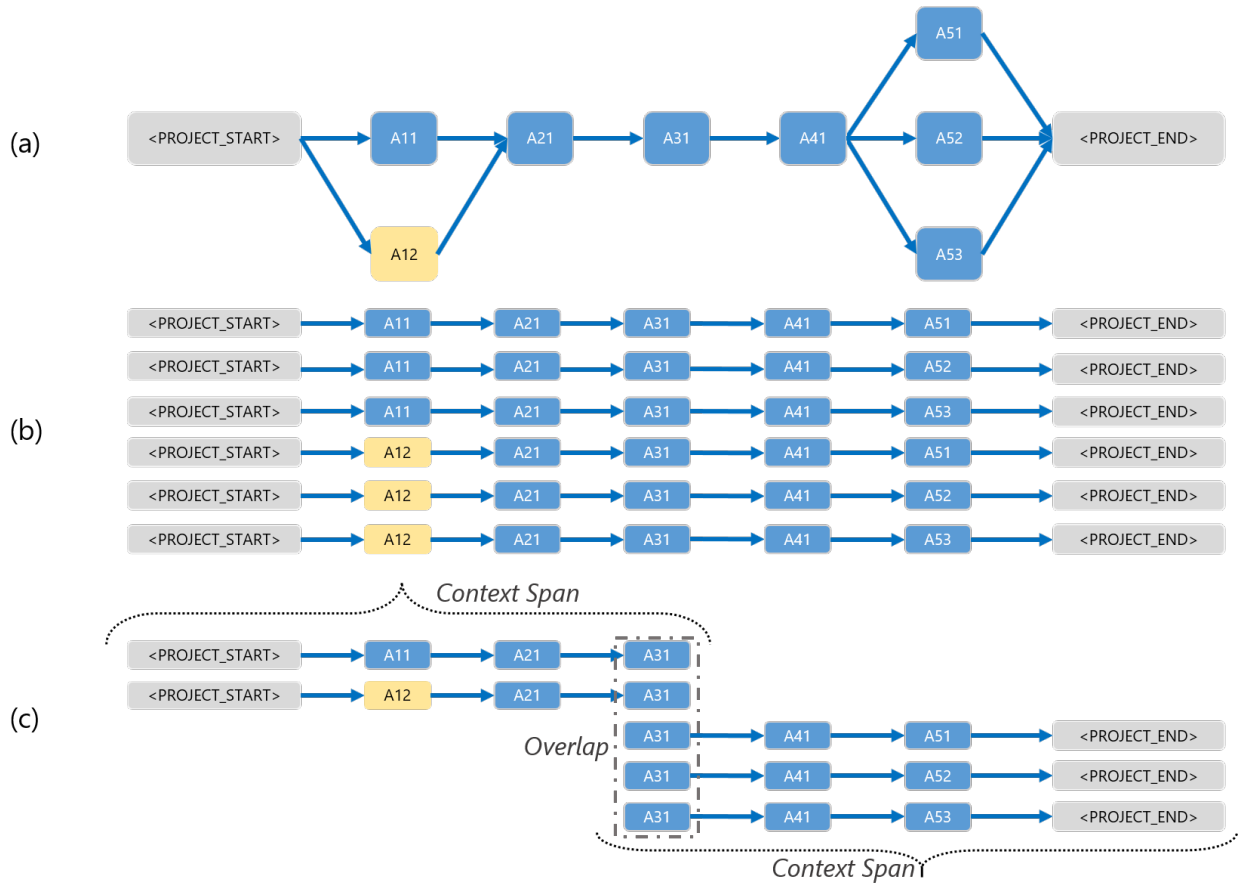


Fig. 3.5. Example on how the number of sequences extracted from a network graph is affected by the sequence length. (a) shows an example extended schedule network after adding one activity. (b) shows the resulting linear sequences if the full sequence length is used. (c) shows the resulting truncated linear sequences for an overlap of one and sequence length of four. The overlap allows the extraction of shorter sequences while preserving available planning information at its entirety.

enables sequential machine learning models, such as the LSTM, to be used for learning purposes. In the following section, we describe how the extracted sequences from schedule graphs along are leveraged together with the learned activity embeddings to train sequencing dependencies for our Dynamic Process Templates.

Dynamic Process Templates (DPTs)

We train two generative LSTM models: One model to learn Successor dependencies and another for predecessor dependencies. As shown in Figure 3.6, each DPT is composed of 4 LSTM layers of size 1600 (5 x embedding size) each followed by two fully connected layers of size 1600 x 1600 and 1600 x number of unique activities (different for each DPT) respectively. The last fully connected layer is followed by a softmax layer to generate a probability distribution over all the possible unique activities. In terms of hyperparameters, each DPT is trained for 150 epochs using RMSProp [161] for optimization, a batch size of 2500, learning rate of 0.0005, and a dropout probability of 0.25 for the LSTM layers. These hyperparameters and model architecture were empirically set after showing the best testing and validation performance. LSTMs were chosen as the main model component given their ability to represent sequential data of different lengths. This criteria enables DPTs to handle the sequential and time-dependent nature of construction sequences while also being capable of accommodating for varying sequence lengths. Such features are not supported by traditional feed forward neural networks [162].

Both DPT_{SUCC} and DPT_{PRED} vanilla architectures are designed to validate our hypothesis that learning planning and sequencing knowledge from existing scheduling records is possible using sequential machine learning models. Here, the activity representation is formulated with Equation 3.1 and the DPT design as functional mappings is described by Equations 3.2,3.3,3.4, and 3.5 below. It is possible to further optimize the presented architectures by fine tuning hyperparameters, adding attention mechanisms [163], or replacing the recurrent LSTM layers with Transformers [164] or other sequential models. Nonetheless, such optimizations were not examined in this paper, as our emphasis is on validating the hypothesis. Our model architecture is considered the first feasible solution as opposed to an end result or a required deliverable.

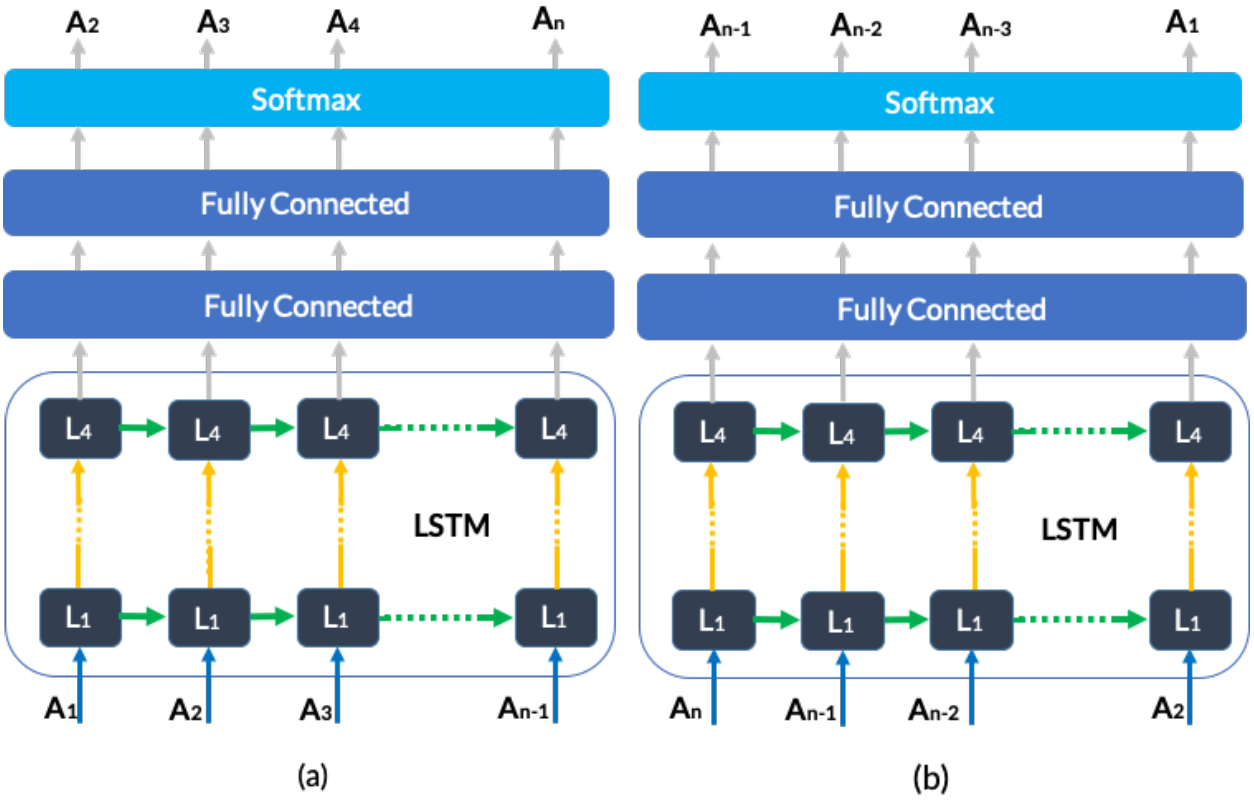


Fig. 3.6. (a) The successor and (b) the predecessor Dynamic Process Templates model architectures and sequence feeding schemes.

The first model $-DPT_{SUCC}-$ is trained on the extracted sequences following the graph direction; i.e., training sequences are passed to DPT_{SUCC} sequentially in a forward fashion (see Figure 3.6a). This model learns a probability distribution over the total number of unique activities available in the data given a sequence of predecessors. It learns to predict the set of most likely successors given an input sequence of predecessor activities. In contrast, the second model $-DPT_{PRED}-$ is trained on inverted sequences; i.e., it is fed with training sequences in a backwards fashion (see Figure 3.6b). DPT_{PRED} learns a probability distribution over the total number of unique activities available in the data given a sequence of successors. It learns to predict the set of most likely predecessors given an input sequence of successor activities. Formally, we let $\mathbb{A}$ to be a fixed set of unique activities of size k_A. Let a be an activity in $\mathbb{A}$ and a_{vec} be its vector representation; i.e., its activity embeddings. Given an activity a in $\mathbb{A}$ at time t and a positive integer k_S defining the length of its precedence sequence, the aim is to learn the following probability distributions:

$$P_{SUCC}(a_t) = P(a_t \mid a_{t-1}, a_{t-2}, ..., a_{t-k_S+1}, a_{t-k_S}) \tag{3.2}$$

$$P_{PRED}(a_t) = P(a_t \mid a_{t+1}, a_{t+2}, ..., a_{t+k_S-1}, a_{t+k_S}) \tag{3.3}$$

Where $P_{SUCC}(a_t)$ is the probability of a_t given a sequence of predecessor activities such that a_{t-k_S} is a direct predecessor of a_{t-k_S+1}, and $P_{PRED}(a_t)$ is the probability of a_t given a sequence of successor activities such that a_{t+k_S} is a direct successor of a_{t+k_S-1}. $P_{SUCC}(a_t)$ and $P_{PRED}(a_t)$ are modeled using function mappings (sequential machine learning models) DPT_{SUCC} and DPT_{PRED} respectively. Given a k_E-dimensional embedding space, DPT_{SUCC} and DPT_{PRED} are defined as follows:

$$DPT_{SUCC} : S^{k_E * k_S} \rightarrow \mathbb{R}^{k_A} \tag{3.4}$$

$$DPT_{PRED} : S^{k_E * k_S} \rightarrow \mathbb{R}^{k_A} \tag{3.5}$$

where $\mathbb{R}^{k_A}$ is a probability distribution of size k_A; i.e., a distribution over all unique activities, and $S^{k_E * k_S}$ is sequence of k_S successors (Eq.3.4) and predecessors (Eq.3.5) where each succes-

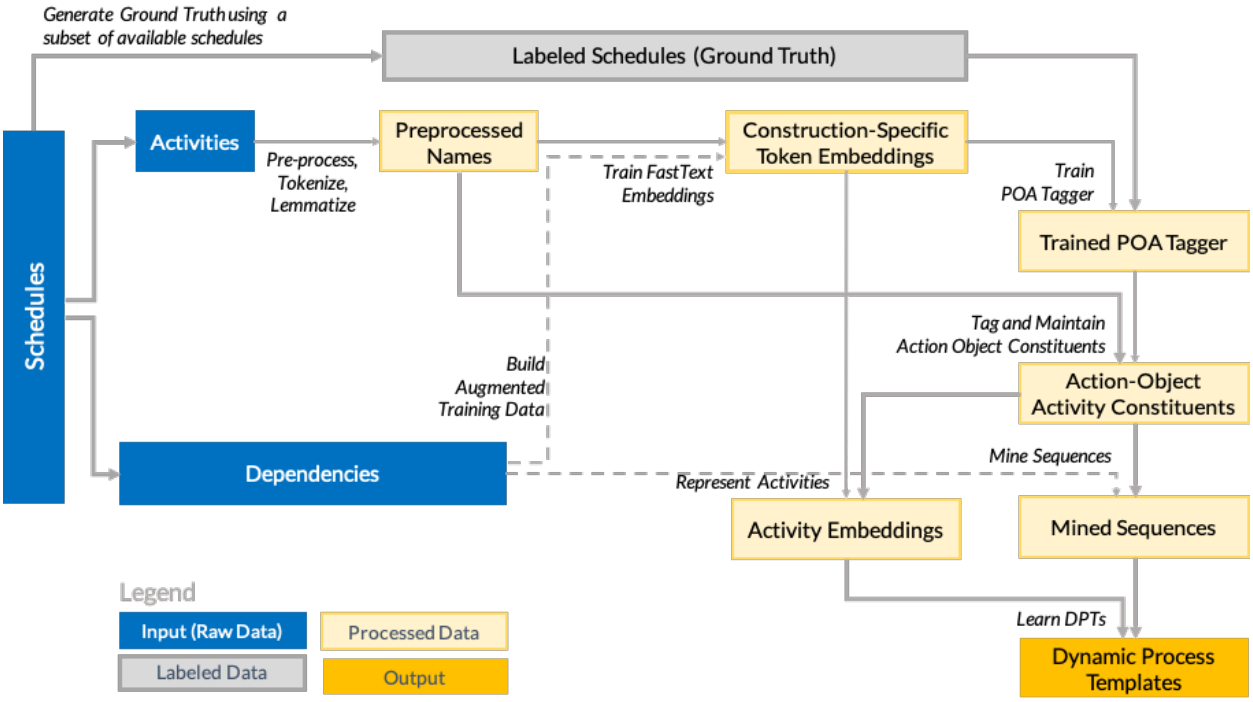


Fig. 3.7. Our overall workflow: *Input*: Raw schedule data in form of *activity descriptions* and their logic dependencies; *Output*: Dynamic Process Templates.

sor/predecessor is vector a_{vec} of size k_E.

In this paper, machine learning is used to (1) model and predict construction activities and (2) model, verify and infer dependencies between successors and predecessors in compact mathematical models. More precisely, machine learning is used to create a mathematical model of the schedule by fitting (1) a function that transforms an activity name into a vector that holds semantic information and allows the identification of similar activities, and (2) two functions that represent the probability distribution of a successor activity given a list of predecessors and the probability distributions of a predecessor activity given a list of successors. This approach supports automated planning and scheduling by providing an easy-access and flexible knowledge database that enables the generation of future plans given the current state of a project and the generation or inference of activities required to produce or revise look-ahead plans.

3.3.3 General Workflow

The overall workflow for learning the DPTs start with taking in raw schedule data as input (see Figure 3.7). As shown, input schedules are parsed for *activity descriptions* and dependency relationships. Each *activity description* is pre-processed and tokenized. The pre-processed terms and their predecessors and successors are used to train construction-specific FastText word embeddings. Those embeddings along with the labeled schedule data are leveraged to train the POA tagger. The tagger is then used to identify the constituents of each activity, and each *activity description* is reduced to tokens referring to *Action* or *Object* constituents. The reduced/generalizable *activity description* along with graph dependencies are used to extract training sequences from the original schedules. Simultaneously, the reduced terms are used to create the set of unique activities along with activity embeddings. Finally, activity embeddings, training sequences, and the set of unique labels are used to train the DPTs. Throughout this process, the only required human supervision is building labeled training data to train a POA tagger. All other processes are performed automatically. This means the process from the perspective of a project planner or an executive is entirely automated.

3.4 Data and Experimental Setup

To validate the method, a schedule dataset from 32 different real-world projects is collected. The scope of these projects, their types, and their sizes vary widely ranging from residential to commercial building projects. Some of the collected schedules have less than 150 activities, and others have more than 10,000 activities as shown in Figure 3.8. Overall, a total of 78,704 individual activities were included in training and testing process. The number of word types for which the model learned construction-specific word embeddings was found to be 2,825. The dataset is therefore composed of raw scheduling data in the form of Oracle Primavera P6 XML files. These files are parsed to extract activity descriptions and dependencies and then processed using the method presented in this paper.

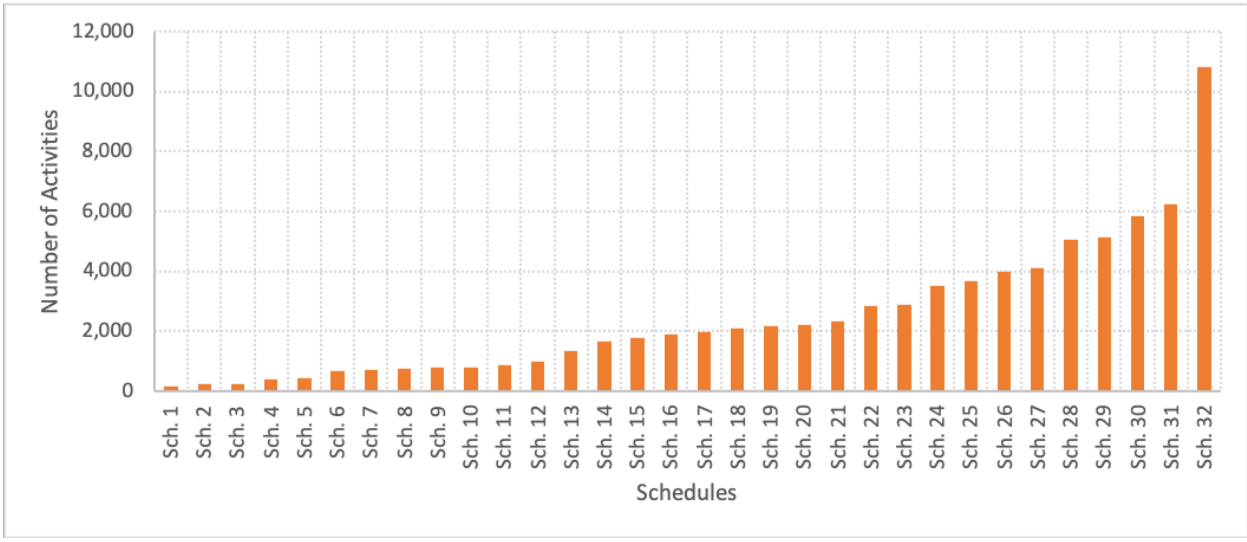


Fig. 3.8. Number of activities in different training schedules.

3.4.1 Word and Activity Embeddings

The size of embeddings was empirically set to 320 which is close to what is typically used for Word2Vec and FastText [156, 158]. Figure 3.9 displays examples on the learned *word embeddings* of "pour" along with its closest neighbors {"slab", "deck", "shearwalls", "form"} after projecting them from their 320-dimensional space into 3D space using t-SNE [165] for visualization purposes using the Embedding Projector [166]. Similarly, Figure 3.10 shows the *activity embeddings* of "pour slab reinforce" and its closest neighbors {"pour slab", "form pour slab", "pour concrete slab"}. These visualizations showcase semantic similarities captured by trained word embeddings and the extension of those similarities to activity embeddings.

3.4.2 Sequence Extraction

Figure 3.11 shows the exponential increase in computational time and the number of extracted sequences as a function of maximum sequence length. To insure the robustness and scalability of the model concerning sequence extraction time and to enable learning sequencing dependencies, we empirically set the context span –sequence length– and the overlap (as defined in Figure 3.4) to four and two respectively. Judging by Figure 3.11, the choice of hyperparameters insures tractable

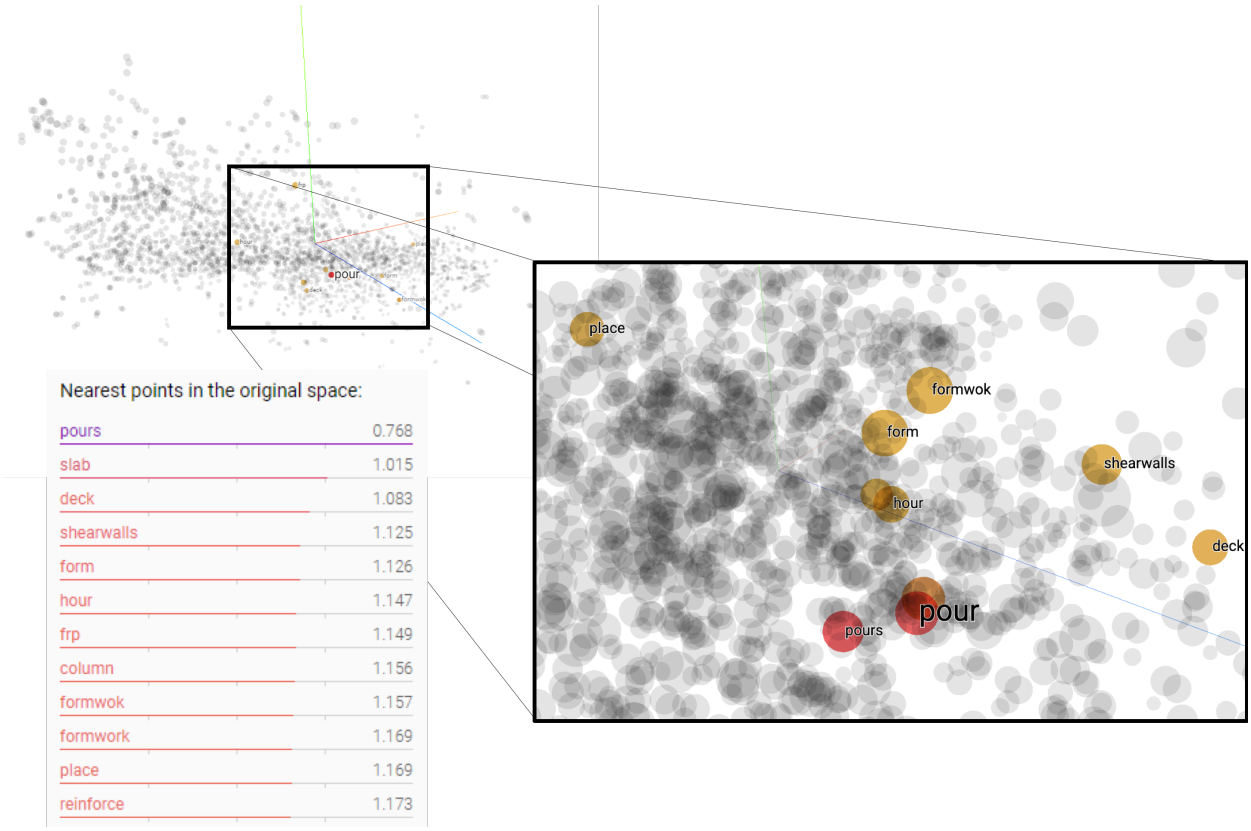


Fig. 3.9. Construction scheduling-specific word embeddings. Example showing the term "pour" and its closest neighbors. The visualization is performed using Google's Embeddings Projector. Readers are encouraged to check the supplementary video of this article for an interactive visualization of the word embeddings.

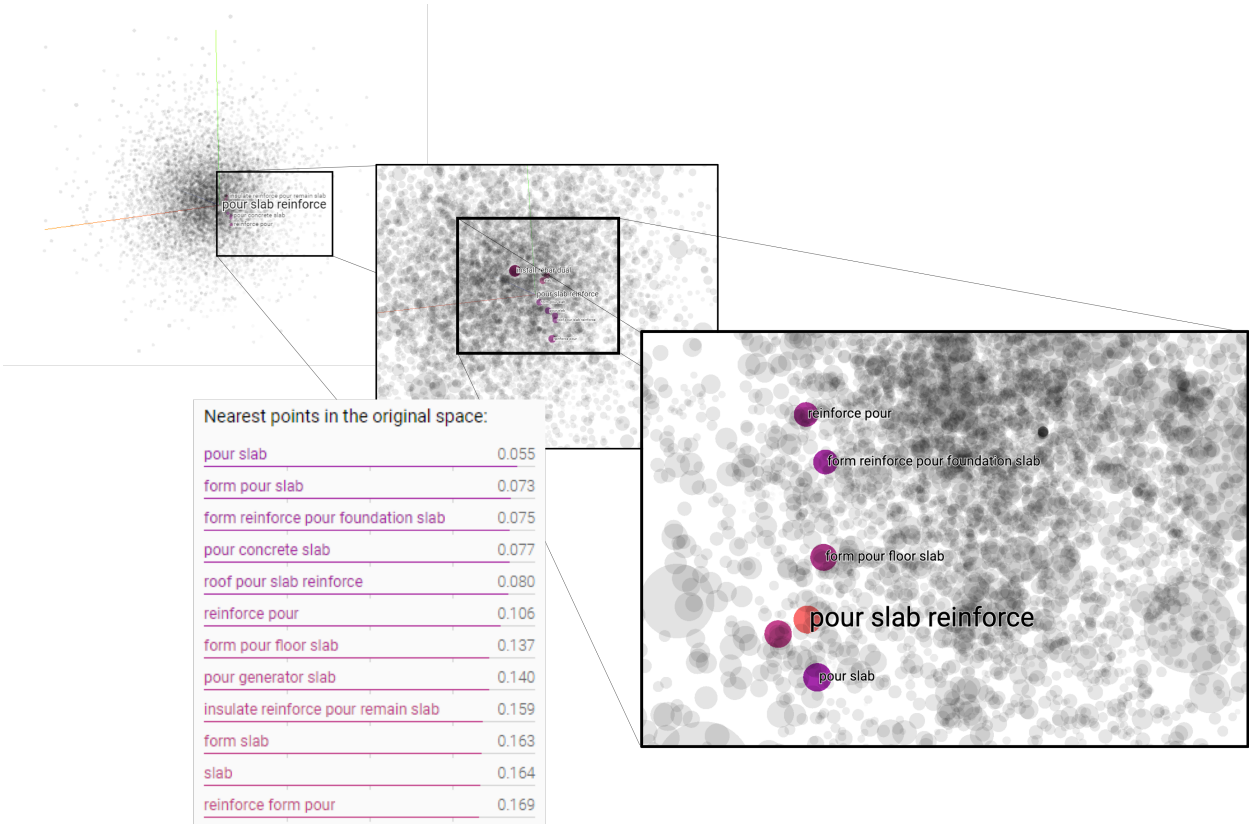


Fig. 3.10. Construction activity embeddings. Example showing the activity "pour slab reinforce" and its closest neighbors. The visualization is performed using Google's Embeddings Projector. Readers are encouraged to check the supplementary video of this article for an interactive visualization of the activity embedding.

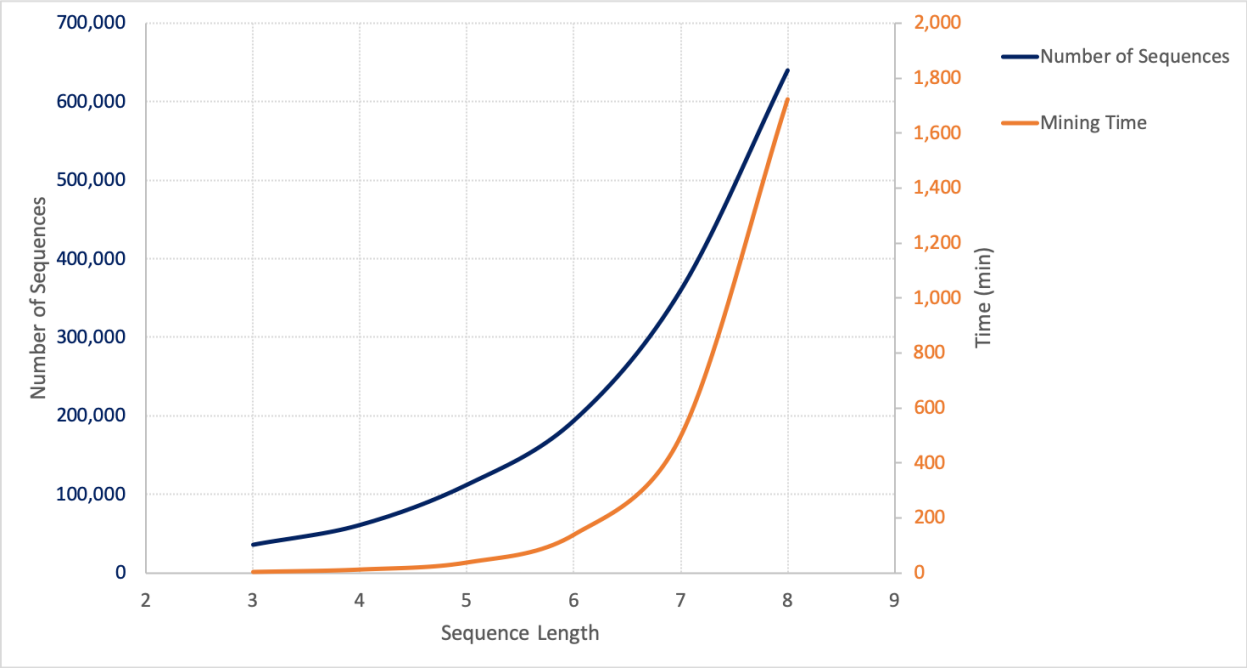


Fig. 3.11. Number of sequences and retrieval time as a function of maximum sequence length for an overlap of 2.

and short computation time. These hyperparameters also condition the prediction of each activity on three consecutive predecessors/successors –ideal for look-ahead planning needs– which allows the models to learn complex sequencing dependencies.

The total number of extracted sequences was 462,316 sequences which were distributed among different projects (see Figure 3.12). As can be seen by comparing Figure 3.8 and Figure 3.12, there is no direct correlation between the number of activities in a schedule and the number of extracted sequences. Rather, the number of sequences in a schedule is dependent on the number of edges connecting different activities (branching factor) and the positions in the graph where these dependencies occur. Also, the number of activities in a schedule does not directly reflect the amount of dependencies that can be learned from that schedule.

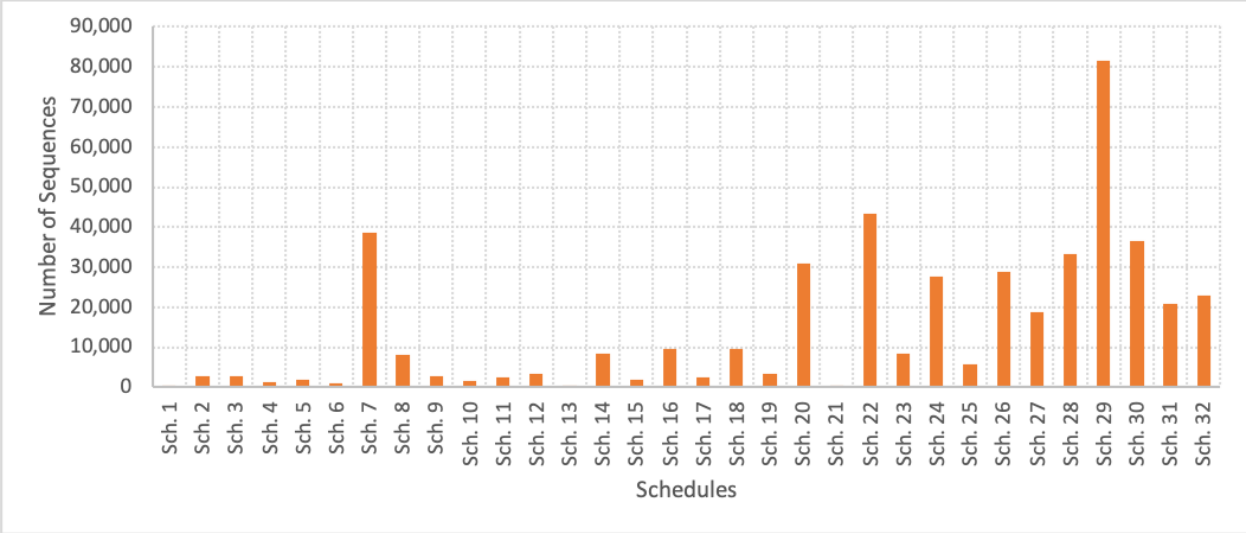


Fig. 3.12. The number of activities in a schedule and the number of extracted sequences are not correlated. Here the number of sequences per different training schedules is shown for sequence length of four and overlap of two between sequences.

3.4.3 Dynamic Process Templates (DPTs)

In order to test the performance of the DPTs, we created four different datasets which are shown in Table 3.2. The "all projects" dataset is composed of all available schedules across different companies and project types. The "single project-type" dataset is built using schedules from "building projects" across multiple companies, and the "company specific" dataset contains multiple project type schedules from a single specific company. Each dataset is randomly split into training (85%) and testing (15%) sequences.

In addition, we created an additional dataset to explicitly test the DPT robustness to the differences in natural language expressions when applied to learned sequencing logic. This "manually modified" dataset was developed by introducing language modifications to *activity descriptions* in one of the schedules that appears in all previous datasets. These modifications include (a) replacing expressions with their abbreviations such as replacing "curtain walls" by "cwall", "mechanical" with "mech", or "air handler" with "ahu", (b) replacing abbreviations with their original expressions such as replacing "F/R/P" by "frame reinforce place", and (c) changing an expression while maintaining its semantic meaning such as exchanging "level k" with "kth floor" or "concrete placement" with "pour concrete". The resulting schedule sequences are extracted and compared

Table 3.2. The number of sequences in available datasets used for training and testing DPTs.

Datasets	Training Sequences	Testing Sequences	All Activities	Unique Activities
All projects	398,155	64,161	78,704	22,900
Single Project-type (buildings only)	162,590	25,920	24,994	9,281
Company-specific (one company)	47,194	7,232	13,990	2,623
Manually Modified *	-	752	-	-

* Sequences extracted after manually modifying *activity descriptions* in an input schedule by replacing expressions with their abbreviations, replacing abbreviations with their original expressions, or modifying an expression while maintaining its semantic meaning.

to the original sequences before inducing the natural language modifications. Only the sequences that did not exist in the original sequences are kept as the "manually modified" dataset.

For each of "all projects", "single project-type", and "company specific" datasets, the authors trained a different DPT composed of both Successor and Predecessor models; i.e., the authors trained three Predecessor DPTs and three Successor DPTs. All models are tested on training, testing, and manually modified sequences. Training and testing sequences are specific to each dataset, and the manually modified sequences used to test all models are the same. This testing procedure is designed to (1) prove DPTs' ability to learn construction sequencing complexities (by computing accuracy on training sequences), (2) test DPTs' ability to predict unseen dependencies (by computing accuracy on testing sequences), (3) examine DPTs' flexibility in handling differences in natural language expressions (by computing accuracy on manually modified sequences), and (4) compare performance between the DPTs based on the scope and breadth of their training dataset.

In any construction schedule, each sequence of activities is expected to have multiple successors/predecessors. We demonstrate the variability in these sequences by computing the average number of successors/predecessors in different datasets (see Table 3.3). The DPTs are not only required to predict one correct successor/predecessor, but they are also expected to predict as many correct successors/predecessors as possible. As such, each DPT must perform a multi-label classification task. Hence, metrics used to test performance are the accuracy of returning the average

number of successors in the case of Successor DPTs and the accuracy of returning the average number of predecessors in the case of Predecessor DPTs. This accuracy is computed at each time step in the input sequence. For Successor DPTs, the model prediction is considered correct at time t when it belongs to the successors of the subsequence starting at time 0 and ending at time $t-1$.

For example, consider the input test sequence *"frame slab"* → *"reinforce slab"*. If our model predicted *"place concrete"* and *"procure doors"* to be the most likely successors and the sequence *"frame slab"* →*"reinforce slab"* → *"place concrete"* appears in the test sequences, but *"frame slab"* → *"reinforce slab"* → *"procure doors"* does not, then the model accuracy on this specific data point would be 0.5. This is true even if *"frame slab"* → *"reinforce slab"* → *"procure doors"* appears in the training data. This is because the aim is to test the model's ability to predict test dependencies that might or might not have been seen previously in the training data. The same logic applies to the Predecessor DPTs, where predecessors are taken into account instead of the successors. A "good model" is therefore defined by the model's ability to correctly predict the expected number of successors/predecessors. If a model returns the average number of successors/predecessors expected in the dataset at a high accuracy, then that model is performing well. The following section presents our quantitative and qualitative results on the DPTs, showcasing the overall performance of the presented method.

3.5 Results and Discussions

We trained Successor and Predecessor DPTs on data extracted from (a) all available projects, (b) building projects only, and (c) a specific company's projects only. As mentioned earlier, the intent is to provide a formal representation for construction sequencing knowledge and leverage it as a resource for on demand querying when scheduling new projects or existing tools that can leverage work templates for schedule optimization purposes. Using machine learning to model DPTs and learn the weights of model parameters through an automated training process, this representation is achieved and validated through attaining high accuracy on training data as highlighted in Table 3.4. This accuracy can be guaranteed and enforced on querying time by designing an interface that

Table 3.3. Average number of task successors and predecessors per task sequence for different datasets.

Dataset	Successors (Train)	Successors (Test)	Predecessors (Train)	Predecessors (Test)
All projects	4.93	2.99	4.87	3.06
Single project type (buildings)	5.81	3.22	5.71	3.22
Company specific (one company)	4.22	2.76	4.12	2.76
Manually Modified *	-	2.76	-	2.58

* Sequences extracted after manually modifying *activity descriptions* in an input schedule by replacing expressions with their abbreviations, replacing abbreviations with their original expressions, or modifying an expression while maintaining the semantic meaning.

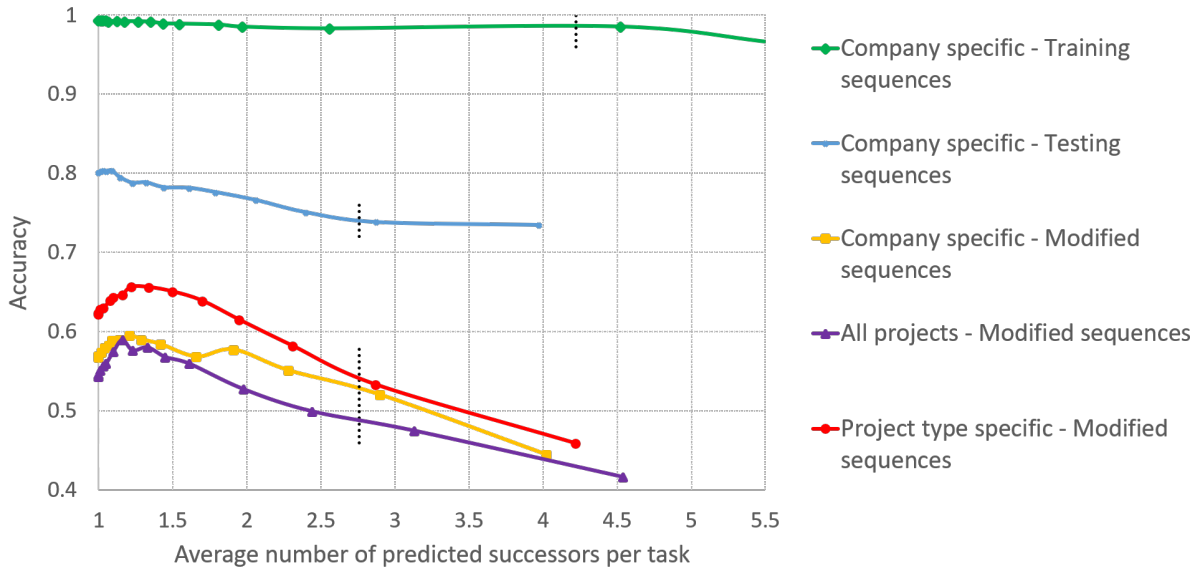


Fig. 3.13. The performance of "company specific" Successor DPTs on training, testing, and manually modified Sequences and "all projects" and "single project-type" Successor DPTs on manually modified sequences measured in terms of accuracy of returning the average number of successors. The dotted lines show the expected number of successors in each specific dataset.

allows users to find and select activities from a dictionary of previously-seen activities.

As shown in Figure 3.13 and Table 3.4, the accuracy on the average number of successors/predecessors is calculated by modifying the prediction threshold of the DPTs. This threshold value defines the minimum confidence required by the DPTs to predict a successor/predecessor. This means the higher the threshold, the higher the confidence in the prediction, but the lesser the number of predicted activities. Since the required task is to return as many correct successors/predecessors as possible, we claim the accuracy on the average number of successors/predecessors in the respective dataset to be a criteria for the model performance. These average numbers of successors/predecessors are shown in Figure 3.13 using dotted vertical lines.

All models were trained for the same number of epochs. For reference, an epoch is one complete optimization cycle through the training dataset specially because the iterative algorithm used for optimizing the neural network requires many epochs during its learning process. Among all models, the ones trained on data from "all projects" have the largest number of parameters. This is because these models perform a classification over a larger number of classes. They are also required to model more dependencies enclosed in their training data due to covering a higher number of companies and project types. Here, being extracted from *multiple project types* means containing different types of logical dependencies, and belonging to *different projects* represents different styles of encoding *activity description* and dependencies into schedules. The task of creating "all projects" models is therefore the most challenging which explains their relatively lower performance.

In addition to validating the method for memorizing sequencing logic, we conducted testing to measure robustness of the models to dependencies that were never seen in the training data. For a Successor model, such dependencies can be split into two categories:

- Category one contains sequences where the successor being tested was never seen as a successor for the input sequence in the training data. For example, if in the training data, sequence "A $\rightarrow$ B $\rightarrow$ C" has "D" and "E" as successors, the presented testing scheme examines the model's ability to predict that an activity "F" seen in the test data is also a successor of "A $\rightarrow$ B $\rightarrow$ C".

Table 3.4. The performance of three different DPTs (trained on company specific, single project type, and all projects datasets) on train and test sequences measured in terms of accuracy (0.0-1.0) of returning the average number of successors for Successor DPTs and predecessors for Predecessor DPTs.

Precedence	Model	Training Sequences	Testing Sequences	Manually Modified Sequences
Successor	All projects	0.76	0.67	0.49
Successor	Single Project-Type (buildings)	0.8	0.63	**0.54**
Successor	Company specific (one company)	**0.98**	**0.74**	0.53
Predecessor	All projects	0.88	0.66	0.50
Predecessor	Single Project-Type (buildings)	0.94	0.63	0.59
Predecessor	Company specific (one company)	**0.98**	**0.72**	**0.62**

- Category two, looks at the influence of partially unseen dependencies on other learned dependencies. For example, if the training data includes sequences "G → H → I→ J" and "G → K → I→ J", the testing scheme would evaluate if the model can predict that "G → L → I→ J" or "M → L → I→ J" include correct dependencies; i.e., such as "I→ J" still being correct for different unseen predecessors.

While category one is unlikely to be predicted by the model because it was never learned, the achieved accuracy on testing sequences shows encouraging results around models' tolerance in handling perturbations in a sequencing logic.

Also, we validated DPTs robustness to variations in natural language expressions with the following question: Given a DPT that "knows" all the dependencies in a given input schedule, how well would the model perform if *activity description* were written differently? In this experiment, almost all models managed to correctly conduct predictions with more than 50% overall accuracy. To get a perspective on the significance of this value, it is worth noting that guessing correctly at random has a probability of 0.027%, 0.07% and 0.23% for "all projects", "single project-type" and "company specific" respectively. The value of this performance should also be compared to

using hard-coded probability tables to model dependencies where activities are stored as strings of words. In the latter case, any unseen test data could only be modeled through smoothing existing probabilities leading to one probability for all explicitly unseen dependencies. By explicit dependencies, we refer to hard-coded string to string dependencies saved as entries in a probability table The DPTs completely alleviate this problem by enabling the handling of unseen data through modeling dependencies using activity embeddings and mapping functions.

To better understand what DPTs actually learn, Table 3.5 is presented. This table shows examples of predictions made by Successor DPT models on manually modified input sequences. The *activity descriptions* in those input sequences are not explicitly seen in the training data, but dependencies between synonymous activities were examined. The same input sequences, examples (a), (b) and (c), are used to sample successors by the models trained on "all projects", "single project-type" data, and "company specific" data.

For the "all projects" model, "place concrete slab" is expected to happen after the sequence "form slab" → "reinforce slab". Although the model specifically predicted that concrete placement should take place on the "roof", which is a POA tagging misprediction, the suggested dependency is correct. Similarly, the model suggested "cure deck" after a "place concrete" activity associated with a "slab" element. This performance showcases the model's ability to draw similarities between "deck" and "slab" as well as learning the logical dependencies between concrete placement and concrete curing. After "f/r/p walls", this DPT suggested forming, reinforcing, placing, and curing the "soffit" or "slab" showcasing its ability to capture cyclical dependencies: "slab → wall → slab → ...".

The "single project-type" model is trained on building projects only. Being specific to buildings, this model managed to provide more insight into the successors of "form slab" → "reinforce slab" when compared to the other two models. For example, this DPT took notice of "install sleeve" and "slab electrical rough in" in addition to "place concrete cure". After placing concrete, the model suggests "stress" highlighting an event in the training data where post-tensioned concrete was used. It also predicts "f/r/p wall", "form install rebar" for forthcoming elements, "perimeter columns", and "install equipment". Similar to the "all projects" DPT, the "single project-type" model was also capable of capturing cyclical dependencies by predicting activities related to the next slab after

"f/r/p walls".

The "company specific" DPT also learned the "form" → "reinforce" → "place" dependency while catching that "sog" (slab on grade) and "slab" as well as "place" and "pour" are synonymous. This model also predicts "set generator" as a successor for placing slab. One can speculate that this could be due to model's treatment of "slab" as a "slab on grade" where a generator is usually placed. This prediction shows how the model is fine tuned towards company-specific practices. Similar to other DPTs, this model learned the "f/r/p cast in place deck" and "deck rebar installation" are successors to placing concrete walls.

Analogous to Table 3.5, Table 3.6 shows examples of predecessors required to enable a sequence of successor activities to occur as predicted by "all projects", "single project-type", and "company specific" Predecessor DPTs. Examples (d), (e) and (f) are also manually modified *activity descriptions* that are not explicitly available in the training data. For example, to enable "install sprinkler", (1) the "all projects" model necessitates "sprinkler main" to be installed, (2) the project specific model requires "painting", "ceiling" and "ceiling tiles" to be installed, and (3) the "company specific" model demands installing "sprinkler main" along with sprinkler "rough in" and mechanical "rough in" to be put in place. Similarly, to enable the sequence: "prime paint" → "install sprinkler", all models predicted that drywalls need to be installed. The buildings specific model further suggested the need for painting materials to be delivered on site before painting can take place. For"tape" and "finish" drywalls *activity descriptions*, all models agree on the need to install drywalls beforehand. Both "all projects" and the "buildings specific" models suggest the whole building be completely enclosed first.

While there still exists cases where predictions are not correct (e.g., predicting the end of a project after walls are placed), all six DPTs presented in this article show insightful recommendations when it comes to suggesting potential successor and predecessor activities. Such recommendations showcase the DPTs' ability to learn logical sequencing dependencies and handle perturbations in both sequencing logic and natural language perturbations.

Our current model ignores differences between Finish-to-Start, Start-to-Finish, Start-to-Start, and Finish-to-Finish relationships, and the DPTs assume all precedence dependencies to be of the

Table 3.5. Qualitative example showcasing the predictive capabilities of the Successor DPTs. Given the same input sequences, each of the three models result in unique predicted set of successor activities.

Model	Ex.	Input Sequence		Predicted Set of Successors
All projects	a	"form slab" → "reinforce slab"	→	{"place concrete *roof* slab"}
All projects	b	"form slab" → "reinforce slab" → "place concrete"	→	{"concrete top out", "form curb", "cure deck *lraw*", "cure deck *lrcw*", **"concrete housekeep pad housing"**}
All projects	c	"reinforce slab" → "form slab" → "place concrete" → "f/r/p walls"	→	{ "reinforce *roof* slab", "*pour* place concrete cure", "*roof* pour install soffit form" }
Single Project-Type (buildings)	a	"form slab" → "reinforce slab"	→	{"*pour* install sleeve", "*pour* place concrete cure", "*pour* slab electrical rough in", **"spray on fireproofing"**, "install mep clip hanger"}
Single Project-Type (buildings)	b	"form slab" → "reinforce slab" → "place concrete"	→	{"install equipment", "*pour* stress", "*pour* f/r/p wall", "roof *pour* f/r/p wall", "form install rebar", "f/r/p wall", "*area b* perimeter col"}
Single Project-Type (buildings)	c	"reinforce slab" → "form slab" → "place concrete" → "f/r/p walls"	→	{**"PROJECT_END"**, "*pour* install soffit perimeter form", "*pour* install sleeve", "*pour* slab reinforce", "*pour* install soffit form"}
Company specific (one company)	a	"form slab" → "reinforce slab"	→	{"pour sog *stair*", "pour sog *ramp*"}
Company specific (one company)	b	"form slab" → "reinforce slab" → "place concrete"	→	{"set generator", "work primary ductbank phase", "curtainwall"}
Company specific (one company)	c	"reinforce slab" → "form slab" → "place concrete" → "f/r/p walls"	→	{ "deck rebar installation", "f/r/p cast in place deck" }

Table 3.6. Qualitative example showcasing Predecessor DPTs' predictive capabilities. The results show the predicted set of predecessor activities by three different models given the same input sequences.

Model	Ex.	Input Sequence		Predicted Set of Predecessors
All projects	d	"install sprinkler"	←	{"sprinkler install main", **"frame inspection"**}
All projects	e	"prime paint" → "install sprinkler"	←	{"tape finish drywall"}
All projects	f	"tape finish drywall" → "prime paint" → "install sprinkler"	←	{"frame", "drywall", "building enclosure complete dry in", "building permit ti"}
Single Project-Type (buildings)	d	"install sprinkler"	←	{"prime paint", "ceiling install", "install ceiling tile"}
Single Project-Type (buildings)	e	"prime paint" → "install sprinkler"	←	{"tape finish drywall", "painting submittals material delievery"}
Single Project-Type (buildings)	f	"tape finish drywall" → "prime paint" → "install sprinkler"	←	{"frame", "drywall", "building enclosure complete dry in", **"door hardware submittals material delievery"**}
Company specific (one company)	d	"install sprinkler"	←	{"sprinkler install main", "above ceiling rough in fire sprinkler pre action", "above ceiling rough in mechanical"}
Company specific (one company)	e	"prime paint" → "install sprinkler"	←	{"drywall finish", "ceiling grid wire install", "drywall installation"}
Company specific (one company)	f	"tape finish drywall" → "prime paint" → "install sprinkler"	←	{"drywall", "drywall installation"}

same type. Handling different types of relationships requires modifications to the current system architecture. In the current architecture design of the DPTs, and to enable fully automated learning, their size and complexities are partially defined by the training data. This could be a limitation for training on a more diverse and significantly larger number of schedule activities. One possible solution to this problem is to cluster similar activities together into a maximum fixed number of activity classes at the model definition phase. Such measures would allow the model to grow only up to a preset fixed number of "activity types", but they are yet to be validated. Clustering could also help the model handle natural language modifications more robustly. Another limitation of the current method is handling different resource dependencies and specially location dependencies. By design, and to enable the model to learn across different projects, we explicitly removed location information from *activity descriptions*. Our current approach enables generality of the presented models, although it limits the model's ability to capture and learn location-dependant workflows. One solution could be to fine tune the POA tagging scheme to enable the identification of a location hierarchy, and then leverage that better explicit location understanding to perform different learning tasks.

In addition to using the DPTs to suggest successor activities and predecessors required to enable a specific workflow, one can leverage DPTS to automatically perform quality control on the logic available in new construction schedules. A solution built on the DPTs, can assess different dependencies in an input schedule and "flag" those *activity descriptions* that seem statistically unlikely. Such a solution provides human planners with a tool for easy and quick review of newly generated project schedules during the pre-construction phase of a project. Finally, the DPTs that "understand" construction sequencing can be employed to help computer vision progress monitoring systems, such as[8], better reason about the status of progress on those elements that have limited visibility or are occluded.

3.6 Conclusions

This article aims at enabling fast and on-demand querying of construction sequencing and planning knowledge by automatically learning the best construction practices from previous projects, and also digitizing them in a flexible and generalizable data schema. To that end, a new method is presented that models activities as vectors of embeddings built using "*Action*" and "*Object* activity constituents. The method leverages this representation to train generative sequential machine learning models to decode logical activity dependencies. The trained models which we call Dynamic Process Templates (DPT) were thoroughly tested on a dataset of 78,704 activities following three different testing schemes. The DPTs proved capable of learning construction sequencing knowledge with accuracy ranging between 76% and 98%. Our results demonstrate the robustness and flexibility in handling unseen logical dependencies and differences in natural language expressions.

CHAPTER 4

LEARNING AND CRITIQUING PAIRWISE ACTIVITY RELATIONSHIPS FOR SCHEDULE QUALITY CONTROL VIA DEEP LEARNING-BASED NATURAL LANGUAGE PROCESSING [3]

[3]This chapter is published in *Automation in Construction* [167]:
"Fouad Amer, Julia Hockenmaier, Mani Golparvar-Fard, "Learning and critiquing pairwise activity relationships for schedule quality control via deep learning-based natural language processing", *Automation in Construction*, Nov 2021, 104036, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2021.104036"

4.1 Introduction

Construction planning is a knowledge-intensive task where planners accumulate data from different sources and in various formats, envision a process to realize the project, and communicate that vision to different project stakeholders in form of a project schedule. The sheer volume of data needed in form of a project scope –usually made available though quantity take-off– and the multitude of the dependencies among activities in forms of structural, functional, legal, resource, information constraints, make planning large projects a difficult undertaking.

According to McKinsey & Company [168], poor planning, among other factors, causes 98% of projects to be delayed by 20 months and to incur 80% increase in cost. To that end, industry guidelines such as those developed by the Project Management Institute (PMI), the Defense Contract Management Agency (DCMA), the Government Accountability Office (GAO), and the National Defence Industrial Association- Integrated Program Management Division (NDIA-IPMD) [169, 170, 171, 172] as well as academic research such as [18, 173, 14] stress the importance of scheduling logic in insuring high-quality schedules and offer procedures and checklists to validate logical dependencies. However, when schedule logic and its impact on the project duration is considered, it is easy to see how small human errors can lead to dire consequences. All too often, mistakes are made by having too many constraints, neglecting trade logic or stacking trades with too many start-to-start and finish-to-finish relationships. In cases where logic is missing, the schedule

has a lot of float embedded in it, which usually results in delays going undetected until late in the project execution phase. The labor-intensive and experience-driven nature of planning can easily impact the quality of a project schedule. Many project-specific conditions can be easily missed. This problem is further exacerbated when the new norm of planning is considered where because of being heavily understaffed, construction companies put one of a few planners in charge of creating schedules for hundreds of projects across their organization.

To help planners create reliable schedules, research such as [62, 75] has focused on automating scope assessments and sequencing, with or without Building Information Models (BIM) as *a priori*. To assess the quality of schedules, and to ensure all the necessary logic is there, research and industry guidelines have also introduced quantitative and qualitative metrics such as establishing a predefined minimum threshold of 90% for the percentage of Finish-to-Start (FS) relationships in the schedule and 5% for "Missing Logical Dependencies" respectively. These methods and metrics are used to describe what constitutes a "good" schedule; e.g., [174, 14, 17, 6]. They are also used to critique pairwise activity relationships, i.e. the individual relationship between pairs of activities. However, they are only developed for a predefined and small subset of activities which are described using a standardized format and with perfect English [17]. For instance, even if the developed method "knows" that the relationship between "*place concrete columns – first floor*" and "*place concrete columns – second floor*" to be a Finish-to-Start relationship, it will fail to judge the relationship between "*place concrete columns – level 1*" and "*place concrete columns – level 2*" because the activities are described differently, and the description text strings do not exactly match those stored in the method's templates. This simplification renders the critiquing solution un-generalizable and inapplicable to real-life construction schedules where activities are described using varying expressions and free form language as shown in [139]. Despite their benefits, scaling these methods and metrics to all projects is challenging because of the semi-structured nature of the wording used in activity description and the lack of standardization. A planner may frame an activity description as *"Place Columns First Floor"*, while another planner may express the same activity as *"Level 1 Verticals Placement"*. The variability in the level of detail necessary for schedules (e.g., master schedule vs. weekly work plans) and how work locations are described across different activities adds to the complexity of schedule quality control as well [139].

To support the need for a systematic schedule quality control, construction researchers compiled

requirements and checklists for streamlining the process of controlling the quality of construction schedules [18, 173, 174]. While existing research stressed the need for checking the logical dependencies among activities, researchers acknowledged the need for human experience and extensive construction knowledge to accomplish that task. Hence, they did not offer any automated solution that can reason about missing or erroneous schedule logic. This task was left for human planers. In this research, we address this specific problem by introducing a natural language processing method driven by a deep neural network to learn and critique pairwise activity relationships based on historical schedule data. Our method accounts for variability in how activity descriptions are formulated in a schedule. To do so, our machine learning model learns sequencing logic from existing records to predict and critique the type of relationship that governs any two input activities. In doing so, our model carefully considers both "activity descriptions" and "Work Breakdown Structures (WBS)" tags when predicting the dependencies between two input activities. This is mainly because the WBS often encodes valuable location information that can improve how activities relationships are evaluated.

Our method predicts both direct and indirect relationships between any pairs of activity description to critique schedule logic. This enables our deep neural model to handle differences in granularity of a project schedule via the formalization and mining of indirect dependencies. The ability to predict indirect relationships over several consecutive tasks also enables us to flag missing activities that are expected to be observed in a work template. We also model the direction of dependencies between input activities. This approach enables our model to digest a list of unordered activities and schedule it using local –activity pair level– pairwise relationship predictions and a global –predicted schedule level– optimization pipeline to maximize predictions consistency. We discuss how our model fits into a more comprehensive schedule assessment method that also implements algorithmic checks on numerical metrics such as duration, lags, or leads. In the next sections, we review the literature of schedule quality assessment metrics and automated methods, we formulate the problem and present our method in detail, and we finally discuss our results and conclusions.

4.2 Related Work

Prior work in automated schedule quality control fall into two categories which are as follows:

4.2.1 Schedule Quality Control Metrics

Over the past decade, several groups have published guideline and best practices to evaluate and control the quality of a project schedule. These works include the Practice Standard for Scheduling from PMI [169], and the guidelines developed by different government agencies such as DCMA's 14-point schedule assessment [170], GAO's Ten Best Practices [171], and NDIA-IPMD's Generally Accepted Scheduling Principles (GASP) [172]. These guidelines outline *logic* as one of their most important criteria for validating schedule quality. For instance, the DCMA emphasizes the importance of insuring all the required dependencies in a construction schedule are present and are accurate. The GAO also stresses that all activities in a schedule should be correctly sequenced and linked based on sound logic.

In addition to practical guidelines by industry practitioners, researchers have also presented multiple metrics for measuring the quality of a construction schedule. Han et al. [18] compiled 49 quality metrics and grouped them into nine categories. In their exhaustive list, the second and fifth categories are targeted at schedule logic for tasks and milestones including relationship types and missing predecessors or successors. Other categories included general criteria such as task names, duration, calendar, constraints, float, leads, and lags. Criteria such as lags, floats, duration can be heuristically checked using computer programs for compliance against a set of predefined thresholds. Nevertheless, checking the logic requires an experienced planner to inspect the schedule and to insure no links are missing or wrongly placed between predecessor and successor activities.

Emam et al. [173] compiled 23 different criteria for schedule assessment purposes. They also validated these criteria through a questionnaire answered by 47 industry professionals. Among various criteria such as constraints and duration, the participants agreed that checking redundant logic, logic density, and types of logical dependencies is essential to building a robust schedule. The inclusion of contractual milestones and implementing the logic enforced by the contract, scored first and third in the survey results in terms of their priority.

Bragadin et al. [14] studied the quality of construction schedules from a process perspective. They introduced a Schedule Health Assessment procedure to help construction planners validate the quality of their schedules compared to a predefined set of heuristics and via color-coding. Their work distinguishes between knowledge of construction scheduling (i.e., means and method of construction) and schedule mechanics (i.e., schedule network development). They make a distinction between the construction knowledge required to sequence activities, and the knowledge required to organize those sequences in a sound schedule. Building on that distinction, their method evaluates activity sequences as a requirement of the schedule mechanics; nevertheless this assessment focuses on the correctness of the network logic and critical path method float and leads. This distinction further highlights a fundamental difference between numerical metrics that can be checked algorithmically, and the metrics that require understanding the semantics of construction operations.

Most recently, Kavad et al. [174] adapted and applied quality control metrics on traditional and 4D BIM-driven schedules. In their work, 22 algorithms are assembled to check 51 numerical metrics such as activity duration and logic density using the Industry Foundation Class schema. Their method evaluates each metric based on a predefined threshold manually set by the user. As such, the method is unable to reason about quality of a schedule as a function of required sequencing dependencies and can not automatically find missing activities. All prior guidelines and schedule quality assessment procedures highlight the importance of correct precedence logic in a project schedule; nevertheless they rely heavily on user experience, user interactions, or manually built-in heuristics.

4.2.2 Automated Methods for Schedule Quality Assessment

In addition to defining metrics, research has also built methods and systems to partially or fully automate schedule quality assessment. One of the earliest work is CRITEX [175], a knowledge-base system that automatically performs construction schedule analysis for mid-rise buildings based on 34 machine interpretable criteria. CRITEX checked a schedule logic using extrinsic rules such as *"An activity that is weather sensitive and not weather-protected should not be scheduled in a period when weather conditions are expected to be below specified minimums or above specified*

maximums". Here, the knowledge-base needs to explicitly model attributes of an activity as well as all logical rules that could influence its sequencing relationship with other activities. For example, an activity must be flagged as "weather sensitive" to be included in the checks performed by the rule mentioned above. As discussed in Dzeng and Lee [176], building such massive knowledge bases can be a tedious task.

Russell and Udaipurwala [177] presented a method for visualizing linear planning charts so that a user can manually check the completeness and accuracy of logical dependencies. Similarly, Bansal and Pal [178, 179] offered methods for visualizing construction schedules using Geographic Information Systems to help human planners manually check the quality of their schedules. The main shortcoming of these expert systems is their inability to learn new rules easily and quickly. To fill in this gap, Dzeng and Lee [176] introduced ScheduleCoach, a rule-based expert system that leverages Case-based Reasoning (CBR) to model and retain the experience of project practitioners. Generally, CBR systems model construction knowledge using case-templates which are composed of $\langle problem, solution \rangle$ tuples. To solve a new schedule evaluation problem, a CBR system first describes the new problem using the predetermined features. Next, it computes the similarity between the new problem and all the problems in the database. The most similar problem is retrieved and its solution is adapted to the new problem. The new problem and its solution are then saved to the database as a new case. Similar to [175], Dong et al. [6] focused particularly on mid-rise buildings and a specific set of "typical" activities. Logical dependencies were modeled explicitly as rules in the form of fragnets [6]. For example, the rule used to describe the process of constructing structural steel is *"The structural steelwork's main sequence should be "steel ordering", "steel shipping", "steel erection", "deck work", "concrete pouring", and then "fire-proof""*. These CBR systems can learn from experience, but require the modeling of a large number of rules. Additionally, designing features that can effectively describe an input problem as well as an adaptation algorithm that can update the solution of an older case to fit the specifics of a new case prove to be difficult.

Among 48 different criteria examined, Moosavi et al. [17, 16] demonstrate that project-specific logic is the most essential and obligatory factor for validating schedule fitness. To validate schedule logic, they developed a "typical schedule" template based on the existing schedules of three building projects. Their template schedule included common activity description, their relationships,

standard duration, and a set of heuristic rules that can be used to compare an input schedule to the "typical schedule". For example, one rule checks whether the duration of electrical installation is within 60% of project overall duration [17]. Their Schedule Assessment and Evaluation system leverages existing data to perform schedule assessment. However, it relied on specific wording such as *"Reinforce"* in input activity descriptions to enable comparison with the template schedule. As demonstrated in [139], activity descriptions are often expressed using a variety of English expressions. For example, *"F/R/P"* often refers to *"Frame, Reinforce, and Place"* in a master schedule. If an algorithm looks to find the specific word *"reinforce"* to identify a reinforcement bar placement activity, then it will fail in cases where this word is not explicitly expressed.

Soman et al. [180] created Resource Description Framework graphs using Shapes Constraint Language to describe and represent construction process information. They presented construction sequencing knowledge using a computer-interpretable format that enables (1) updating look-ahead plans based on progress data and (2) validating the feasibility of the updated plans. Their method evaluates the quality of a look-ahead plan in terms of its feasibility; however, it relies on the constraints already modeled in the schedule and assumes the embedded logic is correct.

Different from all prior work, we present a new method for critiquing the quality of construction schedules, and we focus specifically on precedence logic. Our model learns construction sequencing knowledge from existing records and leverages it to judge the relationships between different input activities using a novel deep learning-based architecture. We use natural language processing (NLP) techniques to enable our model to learn the differences in expressing activity descriptions, and we alleviate the need for using any type of "standardized" or "typical" activity descriptions.

4.2.3 Natural Language Processing in Construction

NLP techniques have been widely applied by construction researchers for the purpose of information acquisition and retrieval [181]. For instance, Zhang et al. used an NLP-based pipeline to recognize and extract semantic and syntactic features from Building Information Models (BIM) and regulatory documents [182], and Kim and Chi used NLP for accident case retrieval and analyses [183]. Similarly, Zhou et al. used NLP to confirm that building designs, in the form of BIM, comply with environmental requirements specified in the contract [184], and Guo et al. leveraged NLP to

extract rules from regulatory documents for automated compliance checking [185]. NLP techniques such as tokenization and part-of-speech (POS) tagging were also applied to map keywords from Requests for Information (RFIs) to Industry Foundation Classes (IFC) attributes and entities [186] to facilitate RFI submittals.

Specific to construction planning, NLP techniques have been applied to learn construction planning and sequencing knowledge from existing records. Amer and Golparvar-Fard [136] built an NLP-based method to identify the ⟨*Action, Location, Object, Responsible Party*⟩ (ALOR) constituents of construction activities based on their unstructured text descriptions. Building on the ALOR schema, Amer and Golparvar-Fard [139] presented a new formalization of construction activities and learned sequencing knowledge from existing scheduling records using Long Short-Term Memory Recurrent Neural Networks (LSTMs). Furthermore, Alikhani et al. [187] leveraged Bidirectional LSTMs to model activity sequences, and Zhao et al. [188] extracted construction methods and logical dependencies from project schedules using POS tagging, n-gram models, and lookup tables. This research builds on the literature of NLP in construction and presents a new method for critiquing logical dependencies between pairs of input activities. The next section formulates the problem and describes the method in detail.

4.3 Method

Construction schedules can be considered as heterogeneous semi-structured graphs where the vertices denote activities and the edges denote direct predecessor-successor dependencies between activity pairs. In a schedule, the activities (graph vertices) are described with descriptions and Work Breakdown Structure (WBS) tags that use plain English words such as "*fix formwork first floor*" and "*Level 1*" respectively. The edges (i.e., the predecessor-successor relationships) are of different types, namely Finish-to-Start (*FS*), Start-to-Start (*SS*), Start-to-Finish (*SF*), and Finish-to-Finish (*FF*) as defined and recommended by PMI [169] and based on common industry practices. The *FS* relationship means the successor activity's start date is governed by the finish date of the predecessor activity, and the same logic applies to all other relationship types.

As discussed in section 4.2.1, there are many aspects for controlling the quality of a construction

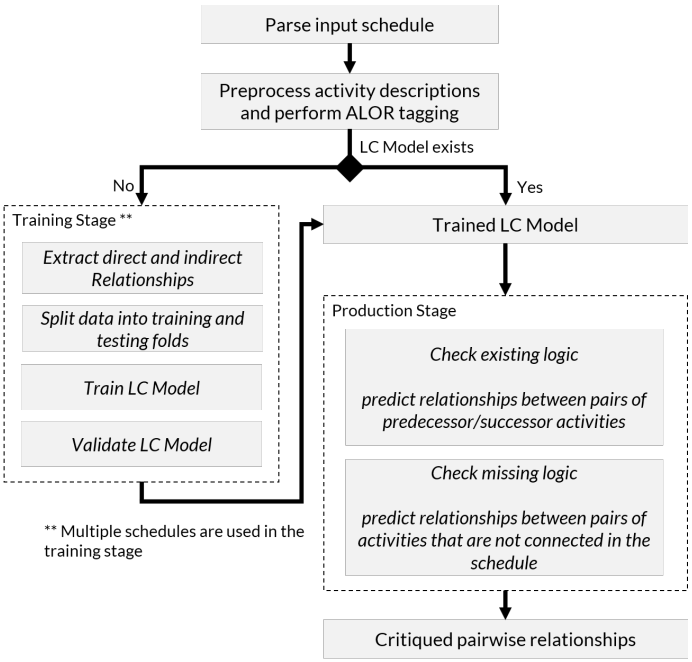


Fig. 4.1. Flowchart showing the overall process of critiquing the logic of an input schedule.

schedule, and one major component of that process is insuring the dependency logic is correct. To assist project planners with validating the logic of their schedules, we introduce a machine learning model that learns and predicts the type and the direction of logic relationship that governs two input activities in a construction schedule. In other words, our method insures all the vertices that should be connected by an edge are actually connected, and they are connected by correct edge type as shown in Fig.4.1.

Our data-driven method is capable of learning a model for logical dependencies and their directions using historical schedule data and without the need for any manually formed heuristics. By learning the direction of a relationship, we mean the model can take any two input activities in any order and predict which one is the predecessor and which one is the successor for the predicted relationship. For example, when feeding our model with *"fix formwork first floor"* as a predecessor activity and *"place concrete first floor"* as the successor activity, our model can predict "Finish-to-Start" as the logical precedence dependency between them. Also, when these activities are fed in a reverse order; i.e., *"place concrete"* as the predecessor and *"fix formwork"* as the successor, our model predicts the dependency to be "Flipped Finish-to-Start" to signal that the input activities should be re-ordered. This capability enables our model to automatically create a

schedule from a list of unordered activities. During project execution, this capability is also helpful as it facilitates measuring maturity of a schedule activity that can start on time for an upcoming coordination meeting, as a function of its required schedule constraints and predecessors. The following introduces the model formulation:

4.3.1 Model Design and Mathematical Formulation

Given *activity description* and *WBS* tags of any two input activities in a construction schedule, our model aims to predict the type of logical dependency that connects them. We denote $\mathbb{A}$ as the set of all possible activities and $\mathbb{D}$ as the set of all possible logical dependencies (see Eq. 6.9). Here, *F* and *S* refer to Finish and Start (i.e., *FS* = Finish-to-Start) respectively, *NR* is no relationship, and the subscript "*f*" denotes a flipped relationship.

$$\mathbb{D} \doteq \{FS, FS_f, SF, SF_f, SS, SS_f, FF, FF_f, NR\}. \tag{4.1}$$

Our model is defined as the Logic Checker (*LC*) function that maps pairs of activities from $\mathbb{A}$ to $\mathbb{D}$ such as:

$$LC : \mathbb{A}^2 \rightarrow \mathbb{D} \tag{4.2}$$

Each activity in $\mathbb{A}$ is defined by an *activity description* and its associated *WBS tag*. Both entities are text strings and are composed of English words. We define $\mathbb{V}$ to be the vocabulary or the set of all possible words that can be used to express an *activity description* or *WBS tag*. Hence, *LC* can be described as the mapping function between $\mathbb{V}$ and $\mathbb{D}$ as follows:

$$LC : \mathbb{V}^4 \rightarrow \mathbb{D} \tag{4.3}$$

In our method, the function *LC* is modeled as a deep neural network. This model is learned from existing scheduling data using supervised machine learning and a fully automated process that requires no additional human annotations other than the information explicitly available in existing project schedules. While labeled data is required for training, training examples and their labels are built by the model completely automatically, using the direct relationships that are available in

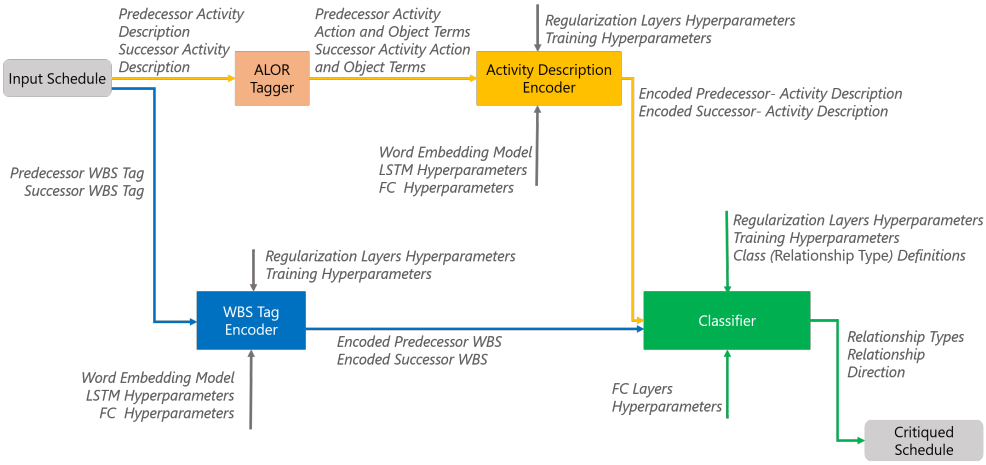


Fig. 4.2. Data and process shown in IDEF-0 diagram representation.

construction schedules and the indirect dependency rules formalized in Section 4.3.2.

Fig.4.2 shows an IDEF-0 representation of data and process in the presented model. This model is composed of an activity description encoder, a *WBS* tag encoder, and a machine learning classifier. The encoders read and encode the information embedded in the description of the predecessor (Fig.4.6 and Fig.4.7) and the successor (Fig.4.8 and Fig.4.9) activities as well as their corresponding *WBS* tags. Next, the encoded inputs –a descriptor of input activities and *WBS* tags– are used as input to the classifier (Fig.4.10). The classifier predicts the type and direction of the logical dependency that governs the relationship between those input activities.

The two encoders and the classifier are based on deep neural network architectures. As shown in Fig.4.3, the encoders are composed of two layers of Long-Short-Term-Memory (LSTM) cells [12] and a fully connected layer that concatenates the output. The LSTM architecture was chosen as the basis of this work to model input activity descriptions and WBS tags. This is because the LSTM architecture can handle input sequences of varying length while traditional feed-forward neural networks [162] do not accommodate varying length. While there exist more advanced models for handling varying length inputs such as Transformers [13], this study was not intended to compare different neural architectures, but instead, our aim is to prove that learning pairwise activity relationships from existing data is possible. The deep neural network architecture and the hyperparameters of the classifier are shown in Fig.4.4. This network is composed of three fully connected layers and a *Softmax* layer with *LeakyRelu* non-linearities and dropouts for regularization

against overfitting. As shown later, the dimensions, hyperparameters, and type of non-linearity were chosen and validated empirically based on model performance.

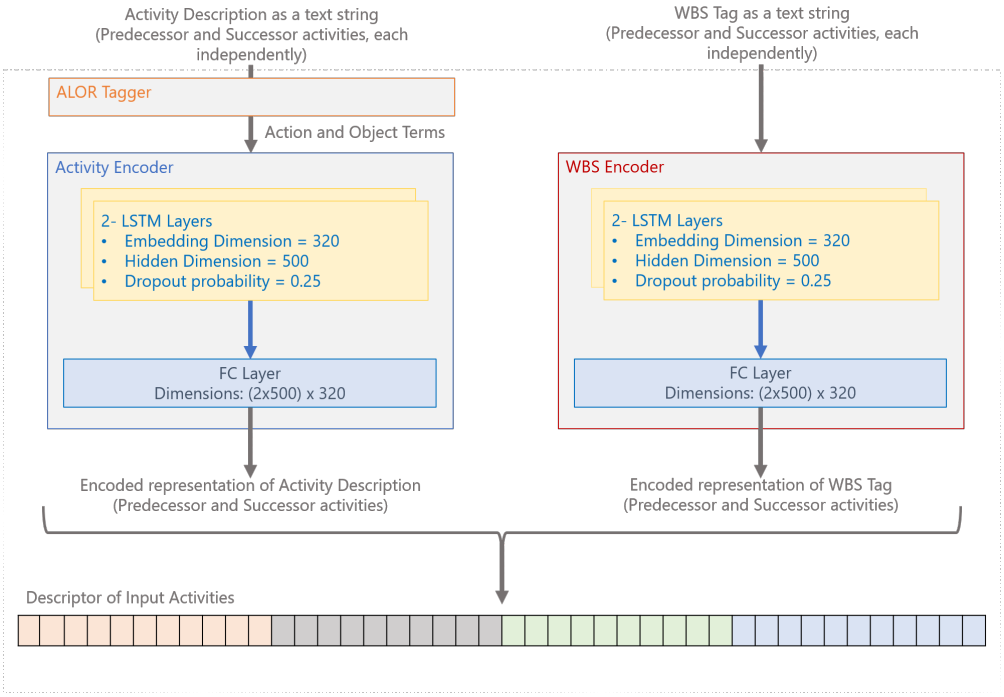


Fig. 4.3. The deep neural network architecture and hyper-parameters of activity description and WBS tag encoders.

As shown in Figures 4.6 to 4.9, the input to our model is a set of two activity descriptions and two *WBS* tags – one set for each activity– per training/testing data entry. Both activity descriptions and *WBS* tags are composed of a number of words from $\mathbb{V}$ –the vocabulary or the set of all possible words that can be used to express an *activity description* or *WBS* tag– that vary among different activities. These words –which can have large variability in how they are written– are projected to an embedding space using the FastText [158] word embeddings model which was developed in [139]. Our embeddings are learned specifically from construction scheduling data as domain-specific embeddings were shown to outperform pretrained off-the-shelf embeddings on

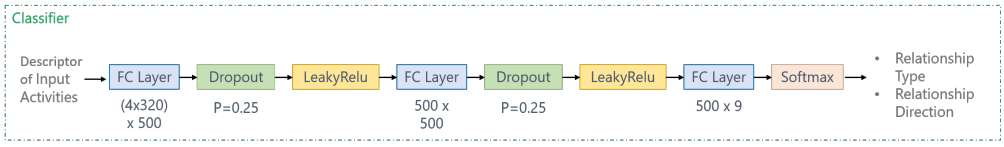


Fig. 4.4. The deep neural network architecture and classifier hyper-parameters for relationship type and direction.

construction related tasks [159]. The *WBS* tags usually contains information about work location and work packages. As such, the *WBS* tags are typically more generic compared to project-specific activity descriptions. This is particularly the case for the higher up levels of hierarchy in the *WBS*. Hence, we chose to maintain the distinction between the two by explicitly feeding them to the model as two separate inputs.

Additionally, we tag activity descriptions using the ALOR tagging method presented in [136]. The tagging model is built using a supervised bi-directional LSTM, and it is trained on manually labeled construction scheduling data. This means, instead of feeding the entire activity description to the Logic Checking model, we decompose the activity description into ⟨*Action*, *Location*, *Object*, *Responsible Party*⟩ tuples and keep the tokens that refer only to an *Action* or an *Object*. As discussed in [139], this process reduces potential noise presented to the model from project-specific information and/or work location names. In Section 4.5, we show the significant improvement in performance when introducing an ALOR tagging step to the Logic Checking model.

Once the ALOR tagging and word embeddings are created, the input is fed to the LSTM cells of the Encoder. These cells maintain a memory on the most important words from the input text string including both activity description and/or the *WBS* tag. Each cell is composed of an input gate, forget gate, output gate, and a hidden cell state. The encoding process for predecessor activity description and *WBS* tags are shown in Figures 4.6 and 4.8. As shown in Fig.4.5, at time t_i, the encoder takes as input the embeddings of word $w^{(t)}$ (word at position t of the input activity description) and the encoder's hidden state after reading the previous input word $w^{(t-1)}$. At time t, the LSTM cells compute new hidden output $h^{(t)}$ and cell state $c^{(t)}$ vectors as follows:

$$i^{(t)} = \sigma\left(W_{ii}w_i + b_{ii} + W_{hi}h^{(t-1)} + b_{hi}\right) \tag{4.4}$$

$$f^{(t)} = \sigma\left(W_{if}w_i + b_{if} + W_{hf}h^{(t-1)} + b_{hf}\right) \tag{4.5}$$

$$g^{(t)} = \tanh\left(W_{ig}w_i + b_{ig} + W_{hg}h^{(t-1)} + b_{hg}\right) \tag{4.6}$$

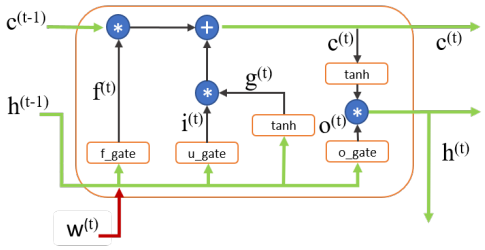


Fig. 4.5. The computations performed by a single LSTM cell at time t.

$$o^{(t)} = \sigma\left(W_{io} w_i + b_{io} + W_{ho} h^{(t-1)} + b_{ho}\right) \tag{4.7}$$

$$c^{(t)} = f^{(t)} \cdot c^{(t-1)} + i^{(t)} \cdot g^{(t)} \tag{4.8}$$

$$h^{(t)} = o^{(t)} \cdot \tanh\left(c^{(t)}\right) \tag{4.9}$$

where $i^{(t)}$, $f^{(t)}$, $g^{(t)}$, and $o^{(t)}$ are internal state variables, tanh and σ (sigmoid) are activation functions, and $\{W_{ii}, b_{ii}\}$, $\{W_{hi}, b_{hi}\}$, $\{W_{if}, b_{if}\}$, $\{W_{hf}, b_{hf}\}$, $\{W_{ig}, b_{ig}\}$,$\{W_{hg}, b_{hg}\}$, $\{W_{io}, b_{io}\}$, and $\{W_{ho}, b_{ho}\}$ are the weights and biases. During model training, these values are learned automatically for different gates and for both input and recurrent variables.

Next, this hidden output $h^{(t)}$ and the next word in the input activity description are fed into the next LSTM cell. As shown in Figures 4.7 and 4.9, the same computation is performed by the *WBS* Encoder to encode the *WBS* tags. The output of the encoding phase is an input descriptor that describes both predecessor and successor input activities. This descriptor concatenates the hidden states of the encoders after reading each predecessor and successor activity description and *WBS* tag. The input descriptor (X_1 in Fig. 4.4) is then fed to the layers of the classifier as follows:

$$X_{i+1} = LeakyRelu\left(W_i X_i + b_i\right) \tag{4.10}$$

$$X_{i+1} = \max\left(0, W_i X_i + b_i\right) - 0.01 \times \min\left(0, W_i X_i + b_i\right) \tag{4.11}$$

where $\{W_i, b_i\}$, and X_i are the {weights, biases}, and input to the i^{th} classifier layer, respectively,

and i $\in \{1, 2, 3\}$.

$$Z = Softmax(X_4) \tag{4.12}$$

where Z is the probability distribution over the relationship types defined by set $\mathbb{D}$ (Eq. 6.9) such that:

$$Z_m = \frac{\exp\left(x_m\right)}{\sum_{k=1}^{k=n} \exp\left(x_k\right)} \tag{4.13}$$

where z_m and x_m are the probability and the classifier's output at a specific relationship type, respectively, and n is the total number of possible relationships which is equal to the length of $\mathbb{D}$. Finally, logical dependency Y is predicted by the model using Eq. 4.14.

$$Y = \text{argmax}(Z) \tag{4.14}$$

As shown in Fig.4.4, during the training of the model, dropout masks with probability 0.25 are applied to X_i to avoid overfitting. In addition to the model architecture, two integral parts of our model are the formalization of indirect dependencies as well as creation on negative training examples; which are discussed in the following sections.

4.3.2 Learning Indirect Dependencies Between Activity Pairs

Construction schedules have different levels of granularity. Also, depending on whether a project is following conventional project controls practices or the Last Planner System [189], they take the forms of master/milestone schedule vs. look-ahead plans vs. weekly work plans. As such, the direct dependencies in one schedule such as using *"erect formwork"* as a predecessor for *"place concrete"*, might not be explicitly available in another schedule, and an *"install rebar"* activity maybe inserted in between the two activities. Our model learns these indirect dependencies to expand the training data beyond the direct dependencies readily available in input schedules. Fig. 4.11 shows 2$^{\text{nd}}$ degree indirect relationships which are formalized for all possible combinations of direct logical dependencies. To our knowledge, this sequencing formalization and learning of

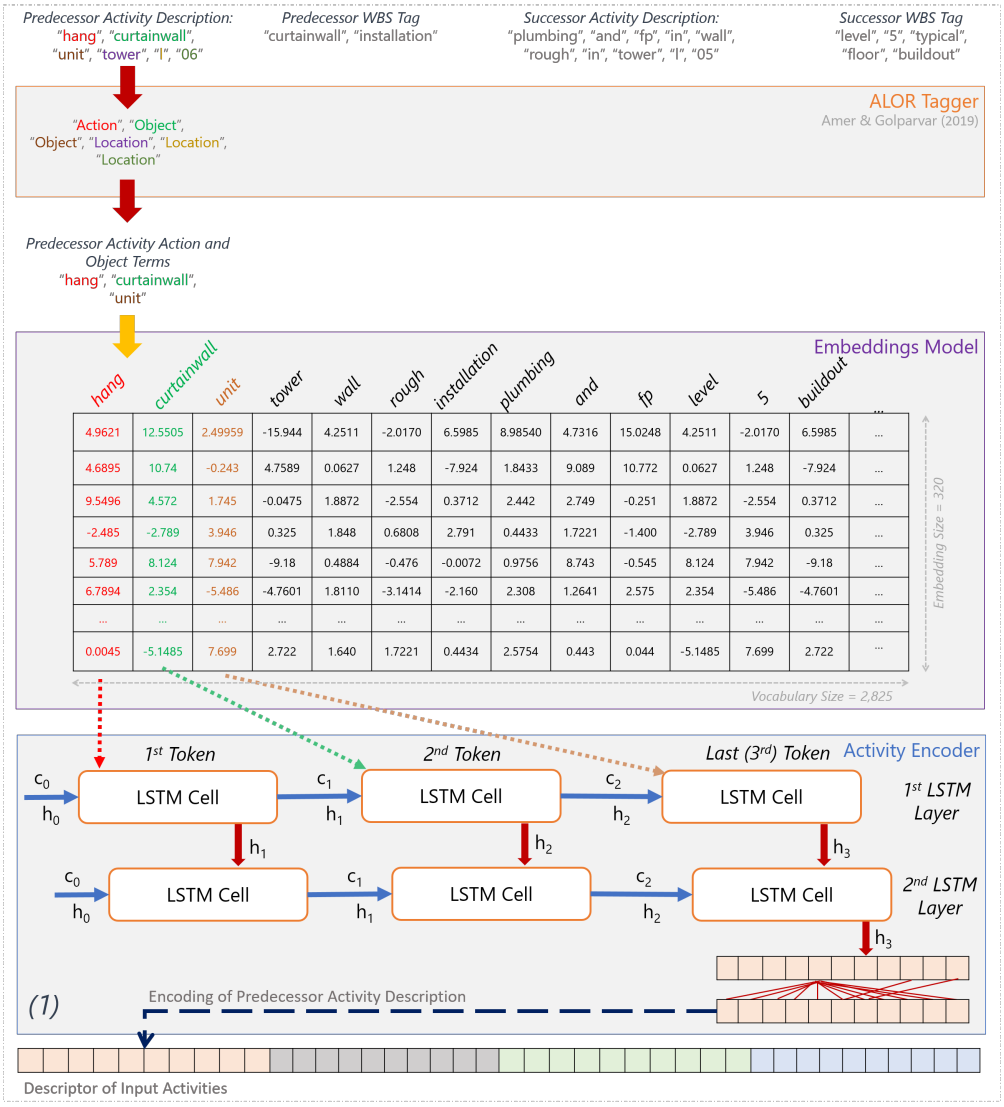


Fig. 4.6. The encoding process for predecessor activity description and WBS tags.

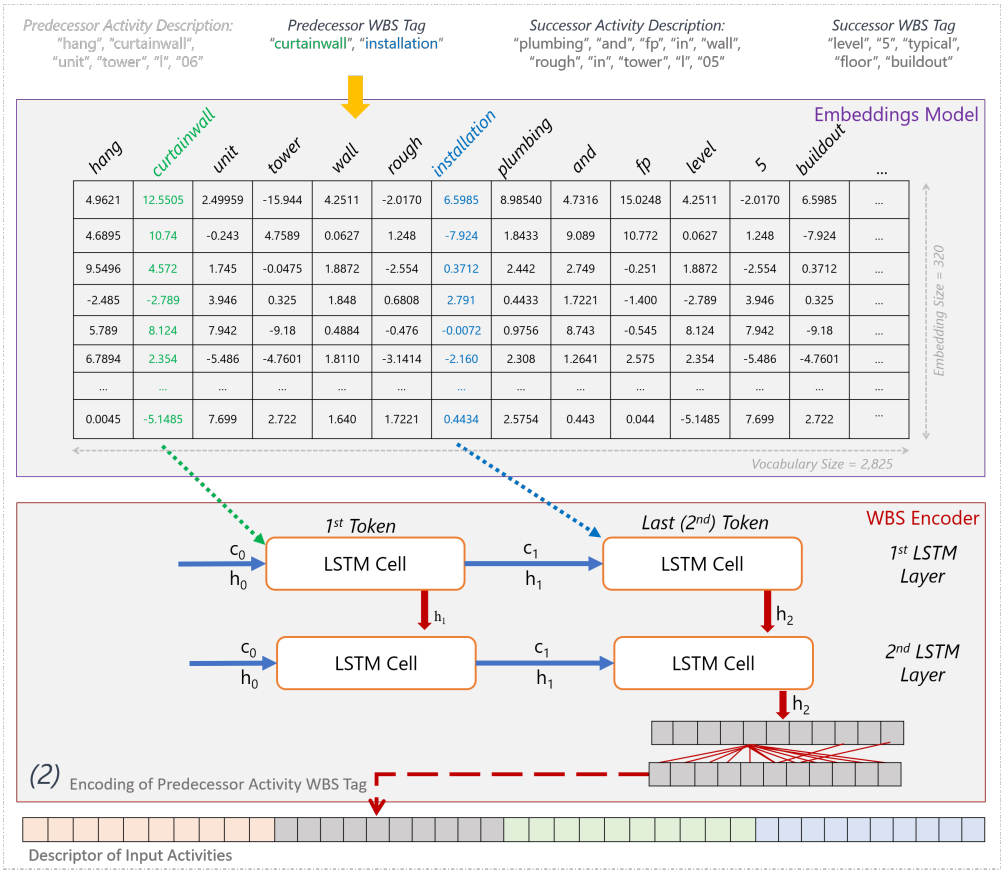


Fig. 4.7. The encoding process for predecessor activity description and WBS tags.

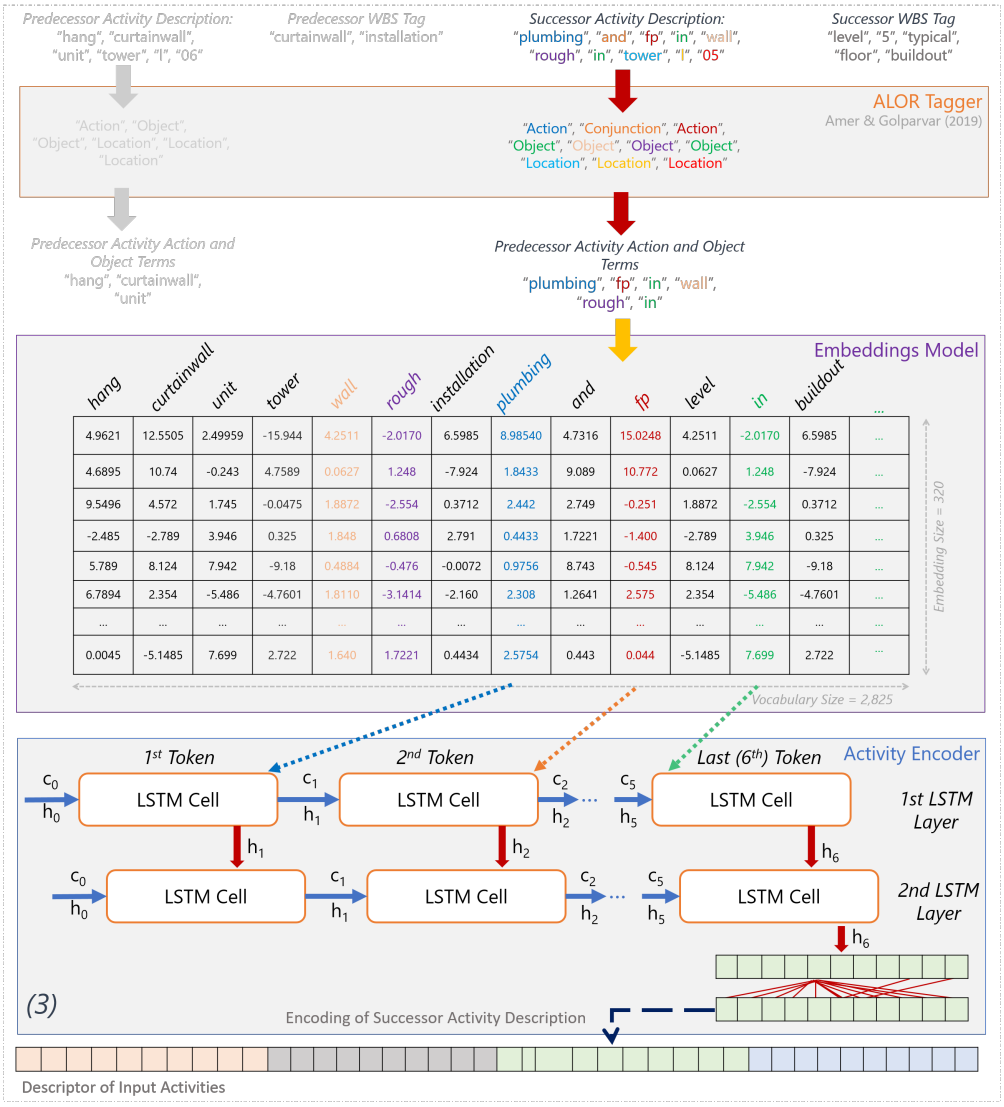


Fig. 4.8. The encoding process for predecessor activity description and WBS tags.

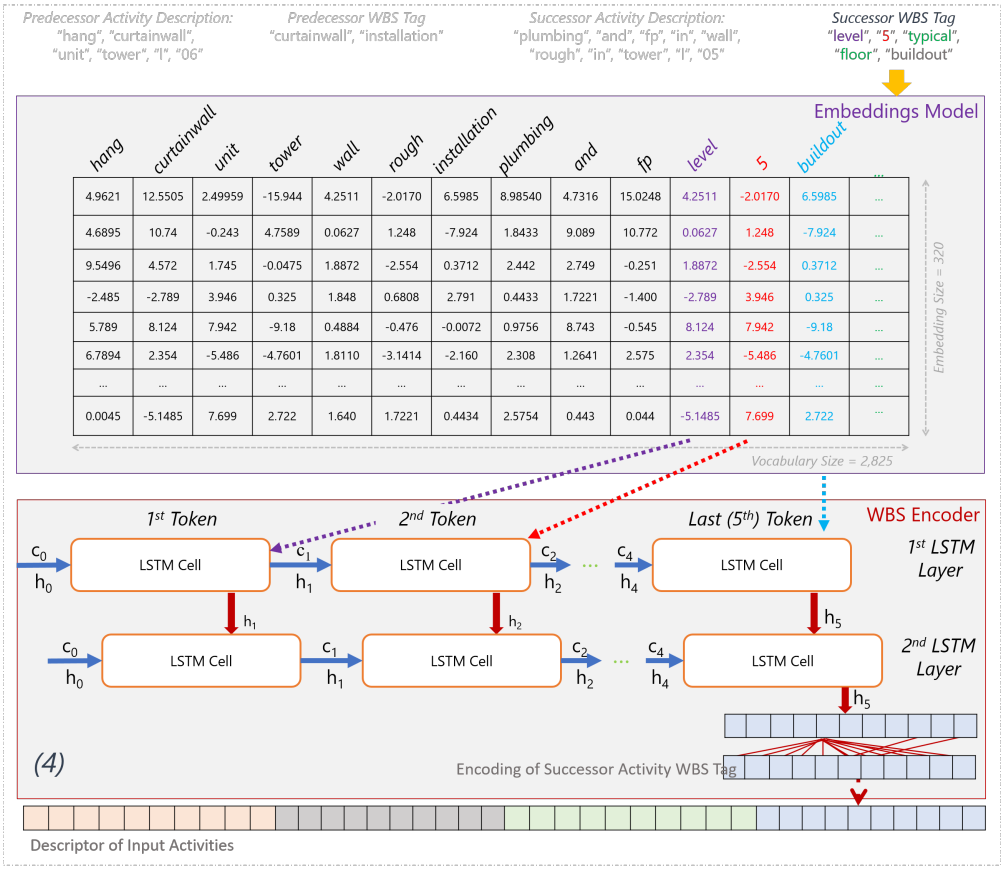


Fig. 4.9. The encoding process for predecessor activity description and WBS tags.

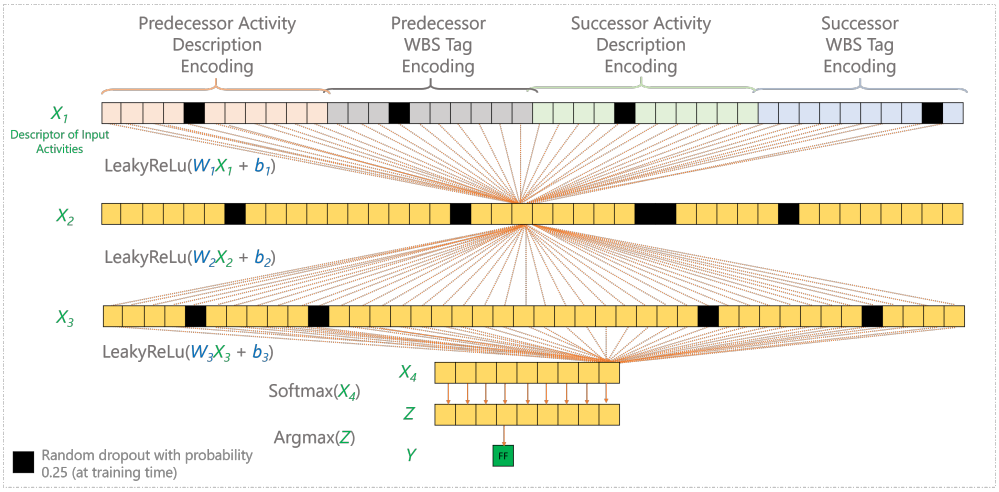


Fig. 4.10. Logic type classification using input from the encoders.

indirect dependencies between construction activities has not been attempted before.

In all examples shown, Activity *A* is the direct predecessor of Activity *B* and their relationship is displayed in Black color, Activity *B* is the direct predecessor of Activity *C* and their relationship is displayed in blue, and the relationship shown in orange is the indirect dependency of Activity *C* on Activity *A* (Fig.4.11 is best seen in color). Qualitative examples of indirect logical dependencies at different depths –defined here as the distance in the schedule graph between two activities– are shown in Table 4.2. We use these formalized indirect dependencies to mine additional training and testing entries from input schedules.

When parsing through a schedule, our model extracts direct and indirect dependencies up to a predefined depth. This depth defines the dependency limit. Without a loss in generality, in our manuscript, this limit is empirically set and validated for up to six indirect dependencies (depth = seven) only. In other words, if a successor activity is seven relationships or less away from a predecessor activity, we consider the two activities to be dependent.

Mining Indirect Dependencies Between Activity Pairs

In our method, an activity relationship *r* is defined as a {*key*, *value*} pair, with *key*={*predecessor, successor*} and *value* = {*Relationship Type*}. After parsing through the schedule, all direct dependencies are represented this representation. Next, we define an Indirect Relationship Mapper (IRM) as a function that maps a pair of direct relationships to their indirect relationship based on the formalization defined in Fig. 4.11 such that IRM: $\mathbb{D}^2 \rightarrow \mathbb{D}$, where $\mathbb{D}$ is defined in Eq. 6.9 .

To mine indirect dependencies, at depth *i*, our method first retrieves direct relationships at depth 1. Next, Algorithm 1 is used to query indirect dependencies at depths 2 to *k*. At any depth *i*, the algorithm searches through relationships at smaller depths that can be merged to create dependencies at depth *i* using IRM (Fig. 4.11). Since indirect relationships between construction activities can be caused by multiple paths in the schedule graph, mining indirect dependencies requires a method to enforce consistency in the mined relationships, which we present in the next section.

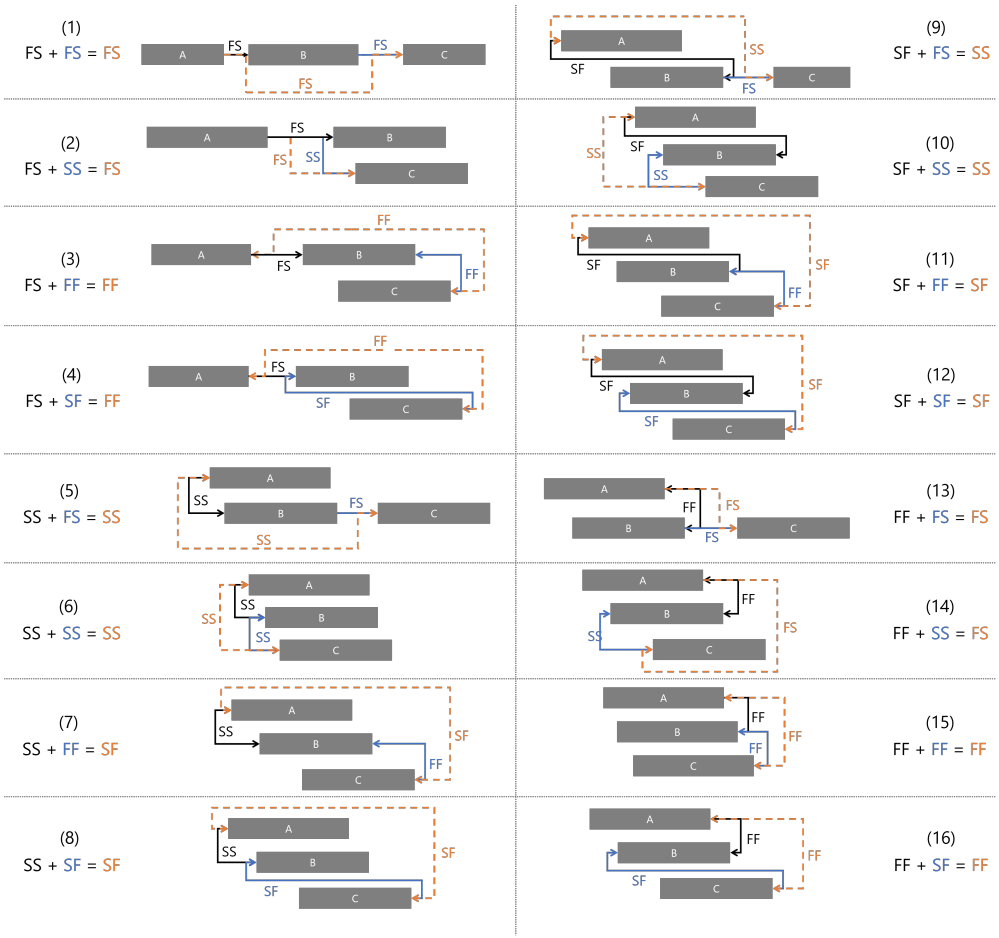


Fig. 4.11. Formalization of 16 forms of indirect activity relationships. Activity A is a predecessor of Activity B and Activity B is a predecessor of Activity C. The arrows in black represent direct dependencies between Activity A and Activity B. The arrows shown in blue are the direct dependencies between Activity B and Activity C. The arrows shown in orange are the **indirect** dependencies between A and C.

Input: $R[1]$ –Schedule's direct relationships at depth 1– where R is the dataset of all schedule dependencies.

Output: Complete dataset R including direct and indirect relationships at depths 1, 2, 3, etc.

for *depth i in [2, 3, ..., k]* **do**
- **for** *(depths m and n) in [1, 2, .., i] such that* $m+n=i$ *and* $m \geq n$ **do**
  - **for** *relationships r1 in R[m] and r2 in R[n]* **do**
    - **if** *r1 successor = r2 predecessor* **then**
      - define new relationship r_{new} at *depth i* such that:
      - r_{new} *key* = (*r1 predecessor*, *r2 successor*)
      - **if** r_{new} *key is not in R* **then**
        - add r_{new} *key* to *R[i]*
      - **for** *all combinations of (r1 value, r2 value)* **do**
        - r_{new} *value* = IRM(*r1 value*, *r2 value*)
        - add r_{new} *value* to the list of possible *values* at *R[i]*[r_{new} *key]*
      - **end**
    - **else if** *r1 predecessor = r1 successor* **then**
      - define new relationship r_{new} at *depth i* such that:
      - r_{new} *key* = (*r2 predecessor*, *r1 successor*)
      - **if** r_{new} *key is not in R* **then**
        - add r_{new} *key* to *R[i]*
      - **for** *all combinations of (r2 value, r1 value)* **do**
        - r_{new} *value* = IRM(*r2 value*, *r1 value*)
        - add r_{new} *value* to the list of possible *values* at *R[i]*[r_{new} *key]*
      - **end**
    - **end**
  - **end**
- **end**

end

Algorithm 1: Pseudocode for mining indirect dependencies from input construction schedules

Consistency of Indirect Dependencies Between Activity Pairs

A pair of activities can have multiple types of relationship based on the graph path used to compute their indirect relationship. For example, given an activity A that has two successors B and C such that (A→B: *FS*) and (A→C: *SS*), and given that both activities B and C have one successor D such that (B→D: *FS*) and (C→D: *FF*). The relationship (A→D) will equal "*FS*", if this relationship was computed following the path A→B→D (Fig.4.11 case #1), and "*SF*" if it was computed following the path A→C→D (Fig.4.11 case #7).

To resolve these conflicts and enable consistent mining of indirect dependencies, we use a priority list of dependency types based on the frequency of appearance in the direct relationships data such that *SF* has the highest priority, and it is followed by *FF*, *SS*, and *FS* respectively. This approach for conflict resolution favors relationship types that appear less often in input schedules and acknowledges that planners only use them when absolutely necessary. The step of conflict resolution is performed after Algorithm 1 is complete. For every relationship *key* in *R*, the *value* of the highest priority is maintained and all other *values* are discarded. Another potential approach to resolving conflicts could have be through voting where the chosen relationship is the one computed following the largest number of paths in the graph, although we did not consider that in our work.

4.3.3 Negative Relationships Between Activity Pairs

In addition to direct and indirect relationships, training the model to predict *NR* relationships (which stands for no relationship as defined in Eq.6.9) requires "negative" training samples. We denote all pairs of activities with no logical direct or indirect dependencies as activities with "negative" relationships. Considering such relationships insures the model is capable of identifying cases where pairs of input activities are expected to be independent. To create a "negative" training set, we search for all pairs of activities in the input schedule that are not identified as directly or indirectly dependent. When a pair of independent activities is found, it is added to the no relationship (*NR*) training set. The total number of required pairwise checks to identify if two activities are linked by a relationship or not is $\frac{1}{2}n(n-1)$ where n is the total number of input activities in an input schedule. This approach guarantees completeness of the data and insures the model learns from all existing and non-existing relationships in input schedules.

4.3.4 Scheduling a List of Unordered Activities

Our model is also designed to learn, infer and form a schedule order for a given list of activities. For any two input activities A and B, our model identifies which activity is the predecessor and which one is the successor, as follows.

Relationship Direction

We introduce a new set of labels: {Finish-to-Start Flipped (FS_f), Start-to-Start Flipped (SS_f), Start-to-Finish Flipped (SF_f), and Finish-to-Finish Flipped (FF_f)}. This formulation enables our model to take activities A and B in any order, predict the logic that governs their relationship, and infer the direction of the relationship. Similar to "negative" relationships, these "flipped" relationships are not readily available in input schedules. To create "flipped" training samples, we augment the data by inverting the original training samples and flipping their labels. For example, if the original schedule contained the relationship {predecessor: *"fix formwork"*, successor: *"place concrete"*, relationship: "*FS*"}, a new training sample is added to the dataset as follows: {*predecessor*: *"place concrete"*, *successor*: *"fix formwork"*, relationship: "FS_f"}.

Schedule Consistency Optimization

For scheduling a list of input activities, our model predicts the pairwise relationship between all activities. Specifically, for every pair, the information fed to the model is local to the pair of activities under consideration. The model does not have the ability to reason about how each pairwise prediction will fit in the final –global– schedule that will join together all the activities in the list. This limitation can cause the model to predict inconsistent dependencies. For example, given three activities A, B, and C, the model can predict that $A \rightarrow B$ is FS, $B \rightarrow C$ is SS, and $A \rightarrow C$ is SS. This prediction is inconsistent according to Fig. 4.11. For it to be consistent, the indirect relationship $A \rightarrow C$ needs to be FS as opposed to SS or $A \rightarrow B$ must be SS as opposed to FS.

To make sure the model predicts a schedule with high prediction scores for local pairwise relationships, we introduce a Constrained Conditional (CC) model [190]. Constraint Conditional models can be used to make sure that classifiers return structured output that obeys certain con-

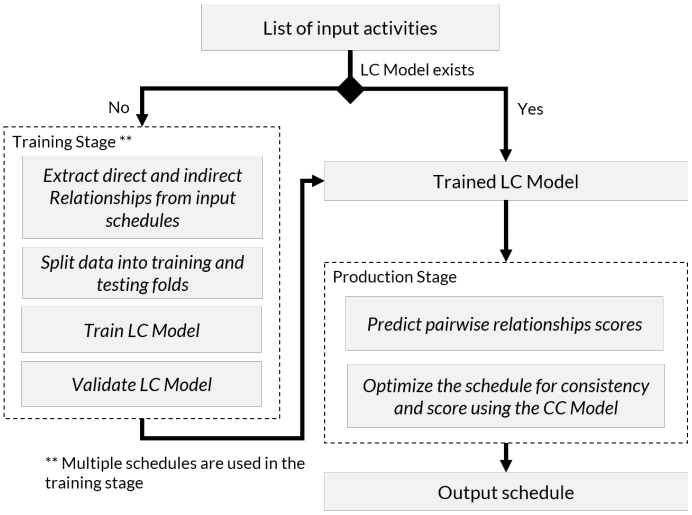


Fig. 4.12. Flowchart showing the overall process of building a schedule from a list of unordered activities.

straints. These constraints are expressed via Integer Linear Programming (ILP), so that an optimal solution to the optimization program consists of predicted outputs that the classifier assigns a high score to and that obey the desired constraints. In our case, the constraints we wish to impose are the indirect dependencies formalized in Fig.4.11. In other words, Instead of merely allowing for the *LC* model to predict the relationship type with the highest score for each activity pair in the input activity list, we want to identify the optimal assignment combination across all pairs of activities that achieves the highest cumulative *LC* model score while causing the least amount of inconsistencies in the final schedule. The overall process of creating a schedule from a list of unordered input activities is illustrated in Fig.4.12. Let Z be the probability distribution over the relationship types defined by set $\mathbb{D}$ as defined by Eq.4.12 and shown in Fig.4.10). It represents the probability scores of each relationship type given the pair of input activities. Z is used as the input to the *CC* model as opposed Y which is the final predictions of the *LC* model (Eq.4.14). Given a list of activities, *CC* models solves the following optimization problem:

$$\mathbf{r}^* = \text{argmax}\left(\sum_{i=1}^{n} Z_{LC}(\mathbf{p}, \mathbf{r}) - \sum_{k=1}^{m} d_{C_K}(\mathbf{p}, \mathbf{r}) \right) \tag{4.15}$$

where n is the number of activity pairs in the input list, $\mathbf{p}$ is the vector of features representing activity pairs, $\mathbf{r}$ is the vector representing relationship assignments to $\mathbf{p}$, which mean an activity

pair can be assigned with any relationship type in $\mathbb{D}$ (Eq. 6.9). Here, $\mathbf{r}^*$ is the vector of optimal assignment of relationships to all input activity pairs, $Z_{LC}(\mathbf{p}, \mathbf{r})$ is the score of assigning $\mathbf{r}$ to $\mathbf{p}$ as predicted by the *LC* model, m is the number of constraints to be checked for consistency, $d_{C_K}(\mathbf{p}, \mathbf{r})$ is a "violation function" that measures the extent of violating a constraint k given a certain relationship assignment configuration $\mathbf{r}$, and it is computed as described in Algorithm 2. The only constraint used for this optimization is consistency with respect to the formalization of indirect dependencies. $d_{C_K}(\mathbf{p}, \mathbf{r})$ therefore checks the amount of inconsistencies created by the model when choosing a specific relationship.

To find $\mathbf{r}^*$, we solve the optimization problem presented by Eq. 4.15, using an A* search algorithm [191, 190]. Here, A* searches for the optimal solution by performing partial assignments of relationship predictions to input activity pairs as described in Algorithm 3 below. In other words, the algorithm starts by assigning only one relationship to one activity pair, and then at every step it expands the relationship assignments to other pairs based on assignment scores that are defined by two functions: (a) an admissible heuristic h that estimates the maximum score that can be achieved by assigning relationships to all the missing activity pairs given the partial assignments of the current partial solution (Eq. 4.16) and (b) a scoring function g that computes the actual score of the current partial solution (Eq. 4.17) taking into account the amount of constraints violation presented by d_{C_K}:

$$h\big(\mathbf{p}, \mathbf{r}_{[1:t]}\big) = \max\left(\sum_{i=0}^{n} Z_{LC}\big(\mathbf{p}, \mathbf{r}_{[t+1:n]} | \mathbf{r}_{[1:t]}\big)\right) \tag{4.16}$$

$$g\big(\mathbf{p}, \mathbf{r}_{[1:t]}\big) = \sum_{i=1}^{n} Z_{LC}\big(\mathbf{p}, \mathbf{r}_{[1:t]}\big) - \sum_{k=1}^{m} d_{C_K}\big(\mathbf{p}, \mathbf{r}_{[1:t]}\big) \tag{4.17}$$

Because we solve a maximization problem, $h\big(\mathbf{p}, \mathbf{r}_{[1:t]}\big)$ needs to never underestimate the score for it to be admissible and to allow A* search to return the optimal solution. This is guaranteed because the inconsistency score provided by $d_{C_K}\big(\mathbf{p}, \mathbf{r}\big)$ is always larger than or equal to 0. In other words, the actual score is always less than or equal to the score estimated by the heuristic function h. This ensures A* never disregards a partial assignment that has potential of being the optimal solution. In the next section, we demonstrate and discuss the data used for training and testing the

presented method.

4.4 Data

To train and test the model, schedule data from three real-world commercial building construction projects from the hospitality sector are assembled. These schedule activities have been carefully reviewed by three expert schedulers, including the project scheduler, for completeness and accuracy. The total number of activities in these schedules is 3,387, and the number and types of direct and indirect relationships are shown in Table 4.1. As discussed in Section 4.3.4, flipped relationships are created by inverting the original relationships mined from the input schedules. Therefore, the numbers of FS_f, SF_f, FF_f, and SS_f relationships are identical to the numbers of *FS*, *SF*, *FF*, and *SS* relationships respectively.

4.4.1 Indirect Dependencies

Table 4.2 shows examples of indirect dependencies associated with a specific predecessor activity: "*level 03 pour b f/r/p columns and walls from level 02 to level 03*" ("*level 03 pour b f/r/p col[umns] [and] wall[s] from level 02 [to] [level] 03*"). Close relationships –i.e., at depth 1 (direct relationships) and at depth 2 and 3– tend to fall within the same WBS branch (e.g., superstructure) and are mostly related to the same type of construction activities or tightly related disciplines. On the other hand, the relationships at depths four and above consist of activities from different disciplines and WBS branches. These examples highlight the hidden dependencies between structural concrete activities and different trades such as envelope (storefronts, windows and eifs), finishing (partitions, tiles, and millwork), and mechanical, electrical, plumbing, and Fire Protection (MEPF) activities (electrical rough in, sprinklers, air handling units). Therefore training over indirect dependencies allows the model to handle different degrees of granularity in input schedules and flag flow disruptions that might not be obvious to a human planner.

Input: List of input (predecessor, successor) activity pairs **p** and a relationship assignment configuration **r**
Output: inconsistency Score
Initialization: Create an empty schedule graph $\mathbb{G}$

Schedule Creation:
for $(r_i, p_i = (a_{ip}, a_{is}))$ *in* **(r,p)** **do**
  if r_i *is a flipped relationship* **then**
    let $a_p = a_{is}$ and $a_s = a_{ip}$ where a_p and a_s are predecessor and successor activities.
    let $r_j = r_i$.flipped() where flipped() is a function that flips a relationship.
  else
    let $a_p = a_{ip}$ and $a_s = a_{is}$ where a_p and a_s are predecessor and successor activities.
    let $r_j = r_i$
  end
  if a_p *is not in* $\mathbb{G}$ **then**
    add a_p to $\mathbb{G}$
    add (a_s, r_j) to a_p successors.
  else
    add (a_s, r_j) to a_p successors.
  end
end

Schedule Pruning:
initiate a seed activity set **s** = {}
let $\mathbb{G}_p = \mathbb{G}$
for *activity* a_i *in* $\mathbb{G}_p$ **do**
  if *a does not have predecessors* **then**
    add a_i to **s**
end
for a_i *in* **s** **do**
  initiate a new list $a_{suc} = a_i$ successors
  for *activity* $a_i s$ *in* a_i *successors* **do**
    for *activity* a_{iss} *in* $a_i s$ *successors* **do**
      if a_{iss} *in* a_i *successors* **then**
        remove a_{iss} from a_{suc}
      add a_{iss} to **s**
    end
    set a_i successors in $\mathbb{G}_p$ to a_{suc}
    remove a_i from **s**
  end
end

Scoring:
let R_{new} be all the direct and indirect dependencies from $\mathbb{G}_p$ mined using Algorithm 1.
compare predictions in $\mathbb{G}$ to relationships in R_{new}.
return violation score = count(inconsistencies) / count(total checks)

Algorithm 2: Pseudocode for computing the violation function $d_{C_K}(\mathbf{p}, \mathbf{r})$.

Input: List of input activity pairs **P** (ex. {(A,B), (B,C) and (A,C)}) and their respective prediction scores **Z** by the *LC* model (ex. Z[(A,B)] = {"FS":0.8, "SS":0.1, etc..})

Output: The optimal relationship assignment in terms of consistency and prediction scores.

Initialization:

Create an empty priority queue $\mathbb{Q}$
Create an empty "visited" list $\mathbb{V}_l$
for *activity pair p in* ***P*** **do**
  for *score of relationship type r of p in* **Z** **do**
    Create a new state s_i that assigns r to p (ex. p = (A, B) and r=*FS*)
    Compute a score for the current state such that score = h(p,r) + g(p,r) (following eq.16 and eq.17)
    Add (s_i, score) to $\mathbb{Q}$
  end
end

Search:

while $\mathbb{Q}$ *is not empty* **do**
  Current state = get state with highest score from $\mathbb{Q}$
  Add current state to $\mathbb{V}_l$
  if *current state has relationship assignments for all activity pairs* **then**
    Return current state as the optimal solution
  else
    Get a list all possible expansions $\mathbb{E}_x$ to the current state (create a list of possible new states by adding a relationship assignment to a missing activity pair from the current solution (ex. add p = (B, C) and r=*FS* to state s_i))
    for *state* s_j *in* $\mathbb{E}_x$ **do**
      if s_j *not in* $\mathbb{V}_l$ **then**
        Compute a score for s_j such that score = h(p,r) + g(p,r) (following eq.16 and eq.17)
        Add (s_i, score) to $\mathbb{Q}$
      end
    end
  end
end

Algorithm 3: Pseudocode for A* Search for finding the optimal relationship assignment $\mathbf{r}^*$.

	Relationship Type				
Depth	*FS*	*SS*	*FF*	*SF*	*NR*
Direct (1)	4,628	396	135	1	-
2	8,489	1,047	266	261	-
3	11,330	2,943	1,126	1,198	-
4	15,284	4,794	2,228	2,197	-
5	17,692	7,240	3,303	2,997	-
6	18,004	11,148	3,813	3,812	-
7	17,097	13,507	4,115	4,431	-
-	-	-	-	-	966,898
Total	92,524	41,075	14,986	14,897	966,898
	8.19%	3.63%	1.33%	1.32%	85.54%

Table 4.1. Datasets of the three real-world commercial building construction projects form the hospitality sector. The Depth factor highlights the degree of closeness between any pair of activities *A* and *B*. For example, a depth of 2 means that Activity *B* is a successor of some Activity *C* which is in turn a successor of Activity *A*.

Predecessor description:	*level 03 pour b f/r/p col wall from level 02 03*							
Predecessor description after ALOR Tagging:	*f/r/p col wall*							
Predecessor WBS tag:	*superstructure*							
Relationship Depth	**1 (Direct)**		**2**		**3**		**4**	
Example	*A*	*B*	*A*	*B*	*A*	*B*	*A*	*B*
Activity Description	*place concrete cure*	*install soffit perimeter form*	*slab reinforce*	*slab electrical rough in*	*storefront*	*install partition frame*	*eifs foam*	*install window*
WBS Tag	*superstructure*	*superstructure*	*superstructure*	*superstructure*	*superstructure*	*superstructure*	*skin*	*skin*
Relationship	FS	SS	SS	SS	SS	SS	SS	SS

Relationship Depth	**5**		**6**		**7**	
Example	*A*	*B*	*A*	*B*	*A*	*B*
Activity Description	*install ceiling frame*	*fire sprinkler install branch*	*install wall floor tile*	*install ahus euhs*	*install ceiling tile*	*install millwork*
WBS Tag	*level 1*	*level 1*	*level 1*	*level 1*	*level 1*	*level 1*
Relationship	SS	SS	SS	SS	SF	SF

Table 4.2. Examples of two successor activities (*A* and *B*) at different depths (0 through 6) for the same predecessor activity: “level 03 pour b f/r/p col wall from level 02 03”.

4.4.2 Training and Data Balancing

As shown in Table 4.1, the data is highly skewed in favor of negative relationships (*NR*). Similarly, among positive relationships, 56.6% of examples have *FS* relationships while *FF* relationships constitutes only 9.17%. To handle the unbalanced data, we use the number of *FS* examples as our reference and we (1) upsample from the *SS*, *FF*, and *SF* examples to match the number of *FS* examples and (2) downsample from the *NR* examples. We train the models for 15 epochs using mini batch size of 1000, learning rate of 0.0005, and ADAM [192] for optimization.

4.4.3 k-Fold Validation

The model is trained and tested following a 5-fold validation schema. The data from all three projects is first randomly shuffled and split into five subsets, and five different models are trained separately: each model is trained on four subsets and tested on the fifth –left out– subset. The final results are computed by averaging over all five models.

4.5 Results and Discussions

We have conducted a set of experiments to analyze the performance of our proposed method. These experiments and their results are discussed in the following sections:

4.5.1 The performance of *LC* model on predicting Pairwise Relationship

Across all possible relationship types, our model achieved an average Precision of 85.14%, average Recall of 95.12%, and average F1 score of 88.3%. These results per relationship type are displayed in Fig.4.13 and the confusion matrix is shown in Fig.4.14(c). Similarly, Table. 6.3 shows the mean and standard deviation of results across all five validation folds. The results show our model accurately learns and predicts different types of logic relationships across unseen schedule test datasets.

The results also show that the model may confuse flipped connections with their forward counterparts as shown by the values displayed in **Bold** in Fig.4.14(c). For instance, the model predicted

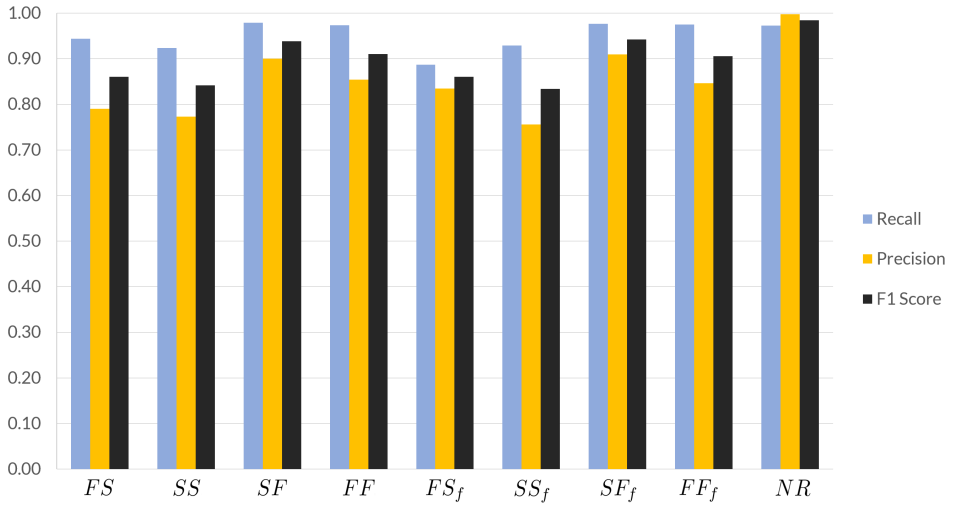


Fig. 4.13. Precision, Recall and F1 Score

	Recall		Precision		F1 Score	
Relationship	μ	σ	μ	σ	μ	σ
FS	0.944	0.53%	0.790	1.64%	0.851	0.88%
SS	0.923	0.64%	0.773	1.93%	0.816	2.34%
SF	0.979	0.18%	0.900	0.31%	0.928	1.08%
FF	0.973	0.48%	0.855	0.95%	0.893	2.04%
FS_f	0.887	1.20%	0.835	1.62%	0.850	1.70%
SS_f	0.929	0.78%	0.756	2.24%	0.801	3.26%
SF_f	0.977	0.17%	0.910	0.16%	0.934	0.99%
FF_f	0.975	0.32%	0.846	1.01%	0.886	2.72%
NR	0.973	0.21%	0.997	0.04%	0.988	0.46%
Overall	0.951	0.50%	0.851	1.10%	0.883	1.72%

Table 4.3. μ (mean) and σ (standard deviation) of Precision, Recall, and F1 Score, across validation folds for the model, trained and tested with ALOR Tagging.

	Recall		Precision		F1 Score	
Relationship	μ	σ	μ	σ	μ	σ
FS	0.952	0.43%	0.711	1.22%	0.814	0.64%
SS	0.940	0.93%	0.684	1.65%	0.791	0.78%
SF	0.971	0.18%	0.716	2.31%	0.824	1.47%
FF	0.975	0.29%	0.719	0.36%	0.828	0.13%
FS_f	0.912	0.24%	0.771	0.69%	0.836	0.50%
SS_f	0.961	0.26%	0.646	0.45%	0.772	0.40%
SF_f	0.974	0.04%	0.745	1.25%	0.844	0.79%
FF_f	0.983	0.08%	0.714	1.54%	0.827	1.00%
NR	0.970	0.00%	0.999	0.00%	0.984	0.00%
Overall	0.960	0.27%	0.745	1.05%	0.836	0.64%

Table 4.4. μ (mean) and σ (standard deviation) of Precision, Recall, and F1 Score, across validation folds for the model, trained and tested without ALOR Tagging.

	Recall		Precision		F1 Score	
Relationship	μ	σ	μ	σ	μ	σ
FS	0.796	2.06%	0.052	0.37%	0.085	0.28%
SS	0.765	3.86%	0.076	1.64%	0.122	2.52%
SF	0.000	0.00%	0.000	0.00%	NA	NA
FF	0.794	7.36%	0.056	1.33%	0.092	3.61%
FS_f	0.691	3.45%	0.070	0.39%	0.110	0.11%
SS_f	0.792	4.73%	0.078	0.68%	0.124	1.29%
SF_f	0.000	0.00%	0.000	0.00%	NA	NA
FF_f	0.804	6.71%	0.066	1.55%	0.109	2.55%
NR	0.947	0.10%	1.000	0.01%	0.834	0.05%
Overall	0.621	3.14%	0.155	0.66%	0.211	1.49%

Table 4.5. μ (mean) and σ (standard deviation) of Precision, Recall, and F1 Score, across validation folds for the model trained and tested on direct relationships only (without any indirect relationships).

(a)

Ground Truth	Prediction: Relationship	Prediction: No Relationship
Relationship	85.89%	0.25%
No Relationship	14.11%	99.75%

(b)

Ground Truth	Prediction: Positive Sense	Prediction: Flipped
Positive Sense	96.85%	1.29%
Flipped	3.15%	98.71%

(c)

Ground Truth	Prediction: FS	SS	SF	FF	NR	FS_f	SS_f	SF_f	FF_f
FS	**79.02%**	4.77%	0.77%	1.98%	0.07%	**0.41%**	1.00%	0.32%	0.25%
SS	1.51%	**77.33%**	0.48%	0.26%	0.02%	0.11%	**1.23%**	0.56%	0.86%
SF	0.03%	0.10%	**90.04%**	1.08%	0.00%	0.00%	0.03%	**0.01%**	0.00%
FF	0.14%	0.05%	0.91%	**85.46%**	0.00%	0.01%	0.06%	0.00%	**0.02%**
NR	**16.18%**	**14.96%**	**6.84%**	**10.11%**	**99.75%**	**14.71%**	**14.94%**	**5.95%**	**10.04%**
FS_f	**2.67%**	1.67%	0.35%	0.25%	0.13%	**83.47%**	6.96%	0.96%	2.87%
SS_f	0.43%	**1.04%**	0.59%	0.82%	0.02%	1.15%	**75.58%**	0.43%	0.25%
SF_f	0.01%	0.04%	**0.01%**	0.01%	0.00%	0.02%	0.13%	**90.98%**	1.09%
FF_f	0.02%	0.04%	0.02%	**0.03%**	0.00%	0.12%	0.07%	0.79%	**84.61%**

Fig. 4.14. Confusion Matrices of the model performance (the final model which accounts for ALOR tagging and indirect activity relationships) on predicting (a) a schedule logic relationship of any type vs. no relationship, (b) a correct vs. wrong logic dependency direction between a pair of input activities, and (c) a correct vs. wrong logic dependency type.

(a) Without Indirect Relationships – With ALOR Tagging

Ground Truth \ Prediction	FS	SS	SF	FF	NR	FS_f	SS_f	SF_f	FF_f
FS	**5.24%**	1.22%	0.00%	1.06%	0.02%	**0.76%**	1.03%	0.00%	0.34%
SS	0.04%	**7.61%**	0.00%	0.05%	0.00%	0.02%	**0.79%**	4.76%	0.36%
SF	0.00%	0.00%	**0.00%**	0.00%	0.00%	0.00%	0.00%	**6.67%**	0.00%
FF	0.02%	0.02%	0.00%	**5.57%**	0.00%	0.00%	0.05%	0.00%	**0.07%**
NR	**93.49%**	**89.59%**	**100.00%**	**92.55%**	**99.95%**	**92.18%**	**88.99%**	**83.81%**	**91.25%**
FS_f	**1.17%**	0.63%	0.00%	0.37%	0.02%	**7.00%**	1.23%	0.00%	1.23%
SS_f	0.02%	**0.80%**	0.00%	0.39%	0.00%	0.02%	**7.83%**	4.76%	0.12%
SF_f	0.00%	0.00%	**0.00%**	0.00%	0.00%	0.00%	0.03%	**0.00%**	0.00%
FF_f	0.00%	0.12%	0.00%	**0.00%**	0.00%	0.02%	0.06%	0.00%	**6.64%**

(b) With Indirect Relationships – Without ALOR Tagging

Ground Truth \ Prediction	FS	SS	SF	FF	NR	FS_f	SS_f	SF_f	FF_f
FS	**71.05%**	4.66%	1.04%	1.76%	0.03%	**0.14%**	0.01%	0.00%	0.05%
SS	1.68%	**68.36%**	1.01%	0.24%	0.01%	0.01%	**0.03%**	0.02%	0.01%
SF	0.06%	0.08%	**71.57%**	1.00%	0.00%	0.00%	0.00%	**0.00%**	0.00%
FF	0.08%	0.02%	1.02%	**71.95%**	0.00%	0.00%	0.00%	0.00%	**0.00%**
NR	**26.63%**	**26.84%**	**25.34%**	**25.04%**	**99.88%**	**21.65%**	**28.60%**	**22.29%**	**24.32%**
FS_f	**0.48%**	0.03%	0.01%	0.00%	0.07%	**77.08%**	6.70%	1.44%	2.81%
SS_f	0.02%	**0.01%**	0.00%	0.01%	0.01%	1.04%	**64.56%**	1.04%	0.32%
SF_f	0.00%	0.00%	**0.00%**	0.00%	0.00%	0.02%	0.08%	**74.52%**	1.04%
FF_f	0.00%	0.00%	0.00%	**0.00%**	0.00%	0.05%	0.02%	0.70%	**71.44%**

Fig. 4.15. Confusion matrix of the model performance (a) without any indirect logic dependencies for schedule activity pairs and (b) without using ALOR tagging for pre-processing activity descriptions.

FS_f while the correct label is *FS* only for 2.47% of the test examples. Similarly, it predicted *FS* when the correct label was FS_f only for 0.41% of the test dataset. These results are similar or better for other relationship types. Overall, as shown in Fig.4.14(b), the precision average error in predicting the sense of the relationship is at most 3.15% (2.22% on average). This highlights that the model learned the direction of the dependencies. For example, the model predicts that schedule activity pairs "*1st coat paint - level 2*" → "*MEP trim out - level 2*" have a *FS* relationship while schedule activity pairs "*MEP trim out - level 2*" → "*1st coat paint - level 2*" have a FS_f relationship.

Most failures of the model relate to predicting *NR*. For instance, the model predicted *FS* for 16.18% of the test examples where the ground truth is *NR*. While this confusion is less than 3% (1-Recall(*NR*)) of all cases, its effect on other labels is still significant given the shear amount of *NR* examples in the dataset, which is a typical problem in highly skewed datasets. Overall, as shown in Fig.4.14(a), the model can accurately predict *NR* relationships in 99.75% of the cases, and it can accurately predict a relationship –of any type– in 85.89% of the cases. Similarly, the model is only 0.25% likely to predict *NR* if a relationship should exist which makes it useful for insuring no important relationships are missed in the schedule.

An ablation study was performed to validate our design decisions regarding (a) the use indirect

relationships for training the model and (2) the use of ALOR tagging to represent construction activities, and the results are shown in Tables 4.4 and 4.5 and the confusion matrices in Fig.4.15a-b. To compute the confusion matrix shown in Fig.4.15(a) and the results shown in Table 4.5, we trained and tested the model using only the direct relationships that are explicitly available in the input schedule. In that scenario, any two activities that are not linked by a direct relationship in the schedule graph will be assumed to have no relationship. Similarly, to build the confusion matrix shown in Fig.4.15(b) and Table 4.4 we trained and tested the model on data that is not pre-processed using the ALOR tagging model. In both cases, the only variable changed in the experiment is the variable being studied (i.e. the removal of indirect relationships in (a) and the removal of ALOR tagging in (b)), and all the other aspects of the training and testing processes were left exactly the same. In Fig.4.15, (a) shows how using direct relationships only (without mining any indirect relationship) causes the model to mis-predict most of the dependencies in favor of *NR* relationships. Introducing indirect relationships improves the model's precision by 74.51%. Also we can see how introducing ALOR Tagging reduced the confusion between different labels and *NR* by an average of 10.65%. This is mainly due to two reasons. First, introducing ALOR Tagging reduces the variability in activity descriptions. For example, both activities "*Place concrete level 2 area b*" and "*place concrete level 1 area a*" will be reduced to "*place concrete*". Since ALOR Tagging was only introduced to activity descriptions, the model still receives information signals about activity locations and packages from WBS tags. This reduction in variability reduces noise and allows the model to better generalize. Second, introducing ALOR tags reduced the percentage of *NR* relationships in the data. This is directly caused by the reduction in the variability of activity descriptions. For example, without ALOR Tagging, if activity "*place concrete level 10*" was eight relationships away from "*form slab level 2*", this relationship would have been added to the dataset as a *NR* relationship even if activity "*place concrete level 2*" was a direct successor of activity "*form slab level 2*" and it was already added as a *FS* relationship. With ALOR Tagging, since both "*place concrete level 2*" and "*place concrete level 10*" activities are reduced to "*place concrete*", the relationship between activities "*place concrete level 10*" and "*form slab level 2*" will not be added as a *NR* relationship because activities "*place concrete*" and "*form slab*" are already linked by a *FS* relationship.

Table 4.6 shows examples of correct and wrong predictions made by LC model, respectively.

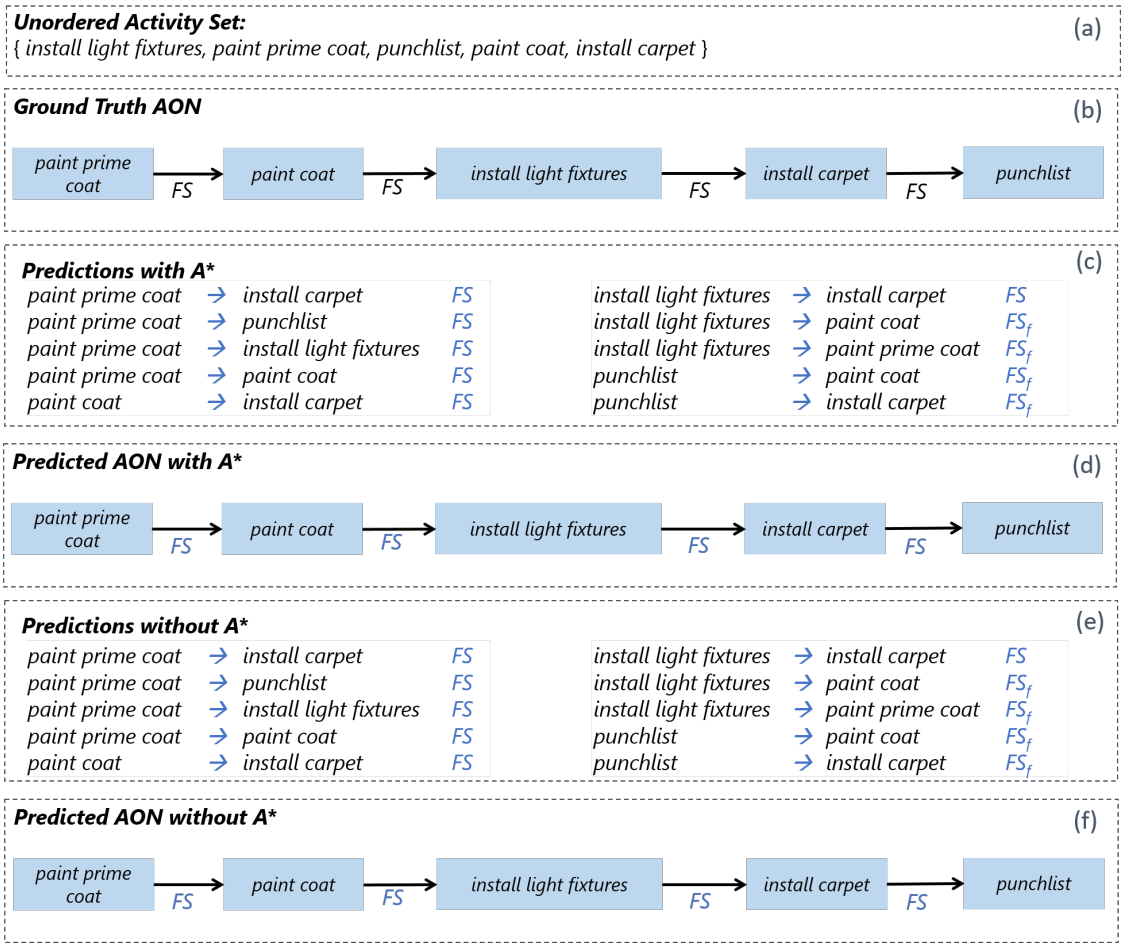


Fig. 4.16. Example showing model performance on scheduling an unordered list of input activities. (a) shows the input list, (b) shows the ground truth schedule (Activity-on-Node (AON) network), (c) and (d) show the predictions made by the model on every pair of activities as well as the predicted pruned schedule using *CC* model, and (e) and (f) show the predictions and the schedule built by the model without using *CC* model. In (c) and (e), the blue arrows highlight consistent dependencies in the predicted schedule and the blue labels highlight accurate dependency predictions.

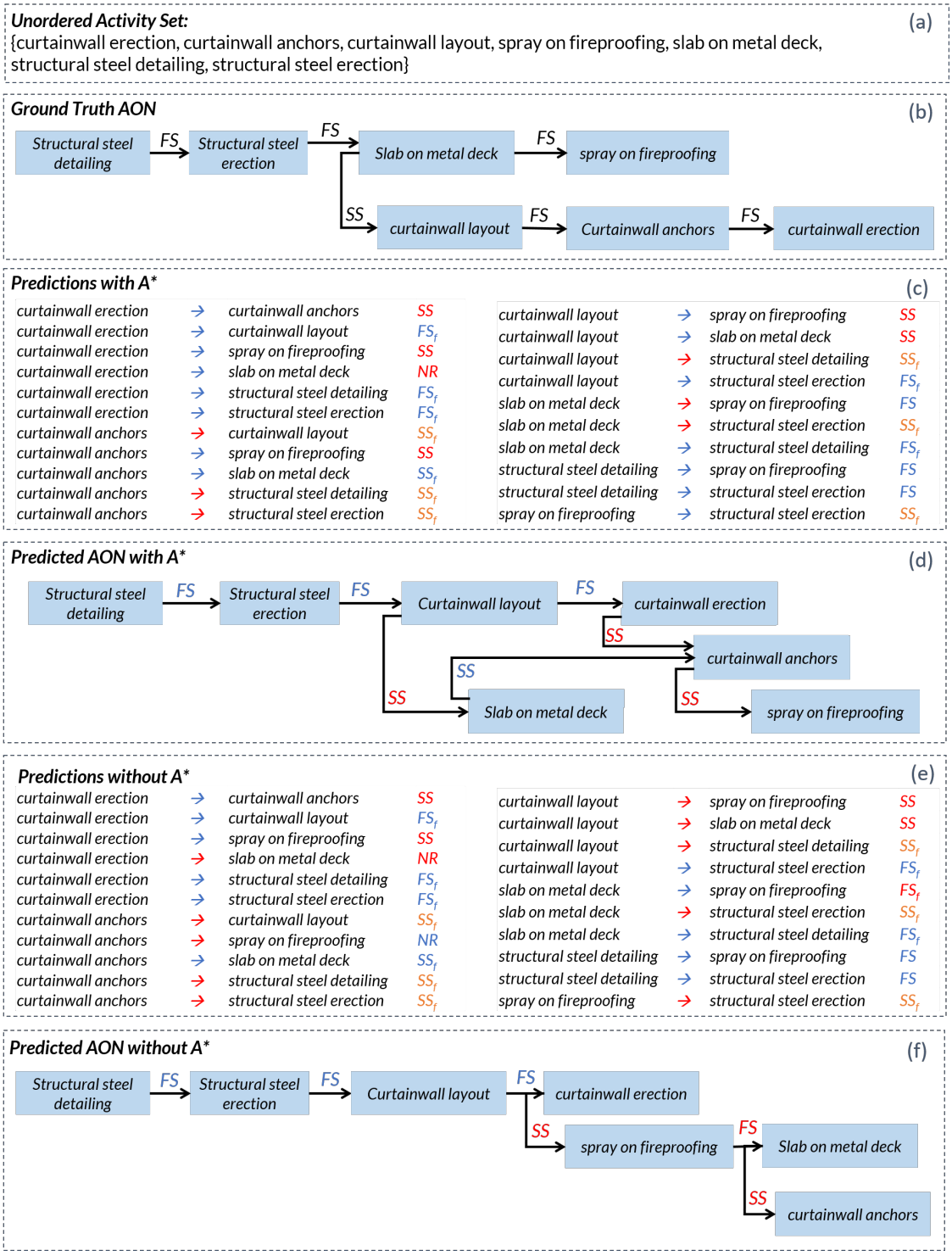


Fig. 4.17. Example showing model performance on scheduling an unordered list of input activities. (a) shows the input list, (b) shows the ground truth schedule (Activity-on-Node (AON) network), (c) and (d) show the predictions made by the model on every pair of activities as well as the predicted pruned schedule using the CC model, and (e) and (f) show the predictions and the schedule built by the model without using CCM. In (c) and (e), blue and red arrows highlight consistent and inconsistent dependencies in the predicted schedule respectively, and the blue, orange, and red labels highlight accurate, accurate in direction, and inaccurate dependency predictions.

Case	Predecessor		Successor		Relationship	
	Activity Description	**WBS Tag**	**Activity Description**	**WBS Tag**	**Ground Truth**	**Prediction**
1a	*sheathe podium*	*podium sheathe*	*stair podium*	*podium stair*	*NR*	*NR*
2a	*subc develop glaze profile approval submittal*	*submittal development*	*erect gate light panel lgt elev w tower crane*	*3 rd 14 th floor*	*NR*	*NR*
3a	*model construction complete incl ff install*	*tier 1 milestone*	*complete mep wall rough in*	*interior buildout*	*FS*	*FS*
4a	*place finish concrete foundation*	*construction*	*install sheet metal mech plumbing ground*	*construction*	*FS*	*FS*
5a	*install block*	*level 1*	*% cd final <UNKN> model*	*design document*	FS_f	FS_f
6a	*complete mep wall rough in*	*interior buildout*	*start form concrete slab*	*construction*	FS_f	FS_f
7a	*install u electrical*	*electrical*	*frame mep rough in*	*buildout*	*FF*	*FF*
8a	*% cd package comment complete*	*detailed design*	*hvac control complete*	*mep*	*FF*	*FF*
9a	*corridor elev lobby install gwb partition*	*level 2*	*drywall bid*	*bid and award*	FF_f	FF_f
10a	*form slab*	*14 th floor*	*strip form*	*9 th floor*	FF_f	FF_f
11a	*install ext frame*	*level 8*	*fan coil*	*level 8*	*SS*	*SS*
12a	*install ext frame*	*level 8*	*guestroom paint prime coat*	*level 8*	*SS*	*SS*
13a	*install ext frame*	*level 4*	*roof pour f/r/p col wall*	*<UNKN>*	SS_f	SS_f
14a	*install cmu masonry*	*1 st and 2 nd floor*	*form slab*	*7 th floor*	SS_f	SS_f
15a	*frame interior partition*	*level 6*	*guestroom install threshold*	*level 5*	*SF*	*SF*
16a	*grind soffit*	*level 3*	*guestroom install wood base*	*level 5*	*SF*	*SF*
17a	*form slab rf*	*roof level*	*form slab*	*10 th floor*	SF_f	SF_f
18a	*install re steel sog*	*area 1*	*install utility tap*	*plumbing*	SF_f	SF_f
1b	*fab deliver elevator*	*fabrication*	*punch list*	*level 1*	*FS*	*NR*
2b	*metal wall panel bid due*	*bid and award*	*architectural panel*	*skin*	*FS*	*NR*
3b	*electrical oh rough in*	*level 3*	*electrical wall rough in*	*level 6*	*SS*	*FS*
4b	*tie pile cage*	*pile installation*	*excavate sand fill pile cap gr beam*	*area 2* a <UNKN> data	*SS*	*FS*
5b	*fire protection standpipe*	*level 7*	*final certificate occupancy*	*c of o*	*SF*	*FF*
6b	*frame penthouse*	*skin*	*project substantial completion*	*milestone and summary item*	*SF*	*FF*
7b	*tower structure*	*hotel*	*remove hoist*	*hoist 2 south*	*FF*	*SF*
8b	*remain excavation*	*er excavation and foundation*	*final commission*	*closeout*	*FF*	*SF*
9b	*frame interior partition*	*level 6*	*grind soffit*	*level 6*	FS_f	*NR*
10b	*elevator lobby install tile floor*	*level 3*	*corridor mep trim*	*level 3*	FS_f	*NR*
11b	*guestroom paint final coat*	*level 7*	*guestroom bathroom tub shower*	*level 7*	SS_f	FS_f
12b	*guestroom install wood base*	*level 4*	*install ext frame*	*level 4*	SS_f	FS_f
13b	*contract project substantial completion*	*completion milestone*	*floor roof structure complete*	*structural*	SF_f	FF_f
14b	*punchlist podium*	*podium punchlist*	*podium envelope*	*hotel*	SF_f	FF_f
15b	*eifs foam*	*skin*	*roof pour slab electrical rough in*	*<UNKN>*	FF_f	SF_f
16b	*final inspection*	*project completion*	*floor form slab*	*3 rd floor*	FF_f	SF_f

Table 4.6. Qualitative examples showing correct (case set "a") and wrong (case set "b") predictions made by the model. The chosen examples highlight the most common mispredictions as shown in Fig4.15

These examples show the wide range of variability of the relationships learned by the model. These relationships have different reasons –logical root causes– and are caused by different flows. For instance, case 5a in Table 4.6 shows an information dependency where installing the floor in level 1 is dependent on finalizing the construction drawings and the building model. While the model does not explicitly learn the root cause of different dependencies, it can implicitly generalize and predict correctly across root causes.

4.5.2 CC model: Scheduling Unordered Activities

As discussed in Section 4.3.4, our model is capable of scheduling a list of unordered activities by introducing "Flipped" relationships to the training data. These new labels allow the *LC* model to digest two input activities in any order and predict the direction and type of dependency that governs their relationship. Similarly, we introduced the *CC* model to enable our model to reason about the consistency of the predicted schedule as a whole and to create a schedule with the highest score of local –pairwise predictions– and the least amount of inconsistencies in indirect relationships. Fig.4.16 and Fig.4.17 show two qualitative examples for creating Activity-on-Node (AON) networks –schedules– from unordered lists of activities. Fig.4.16 shows the scheduling of five test activities: {*install light fixtures, paint prime coat, punchlist, paint coat, install carpet* } which are all linked through *FS* relationships as shown in 4.16(b). By looking at 4.16(c) we can see that our deep learning based *LC* model accurately and consistently predicted all pairwise relationships. Since the *LC* model predictions are consistent, the optimization framework –*CC* model– also predicted the relationships with the highest prediction scores. On the other hand, Fig.4.17 shows the results for scheduling test activities: {*curtainwall erection, curtainwall anchors, curtainwall layout, spray on fireproofing, slab on metal deck, structural steel detailing, structural steel erection*}. In 4.17(c) and 4.17(d), the color of predicted relationship notes whether the prediction is accurate (blue), accurate in terms of dependency direction (orange), and inaccurate (red). On the other hand, the color of the arrows show whether the predicted dependency is consistent with the created schedule (blue) or not (red). By comparing the number of red arrows in 4.17(c) (6 inconsistencies) to those in 4.17(d) (10 inconsistencies) it can be seen that the *CC* model decreased the number of predicted inconsistencies, hence creating a schedule that is globally more coherent and more accurate. The

accuracy of predicted relationships in the final schedule is 4/7 for the *CC* model while being 3/6 for the *LC* model (ratio of blue to total relationships in (c) and (d)). The accuracy of the schedule predicted by *CC* model is dependent on the scores assigned to different labels produced by the *LC* model: the better the *LC* model performance, the better the overall prediction by the *CC* model.

Our formulation of the *CC* model decouples it from the predictor used as a *LC* model. Future research can improve on existing results by implementing more advanced neural architectures such as Transformers [13] to improve the local pairwise prediction by the *LC* model. Similarly, the formulation of the *CC* model allows for incorporating additional constraints to create a final schedule from a set of *LC* model scores. While in this paper we introduce only one $d_{C_K}(p_i, r_i)$ (Algorithm 2) for optimizing the consistency of the created schedule, future research can expand on this formulation by introducing time, cost, or other optimization parameters.

In terms of limitations, since our model is completely data driven, it learns the biases and the mistakes that exist in the training dataset. However, when training on large datasets, we expect the model to correct for errors of individual human planners. It is unlikely for all planners to exert the exact same mistake, but it is likely that they all use the same correct logical dependencies. Similarly, we currently set the depth of indirect dependencies empirically to six as discussed in Section 4.3.2. This hard threshold can lead to errors in the training data where some activities are assumed to be indirectly dependent while in fact they are not. This is specifically the case when the input schedule is not very detailed. On the other hand, if the schedule is too detailed, then some activities that are more than six indirect relationships away can be also dependent and are currently assumed independent by the model. More studies can be conducted to better understand the effect of specifying a dependency limit and to better label the dependencies mined from input schedules. Additionally, irrespective of the dependency limit, the training data will be skewed. This unbalancing in the training data is enforced by different scheduling guidelines such as [170] and [171] where the percentage of "irregular" relationships –*SS*, *SF*, and *FF* – is bound by a small threshold by design.

In terms of scope, our method is only focused on validating logical dependencies. It does not provide a general framework for assessing the overall quality of a schedule, and it is designed to be part of an overall critiquing system that also checks the other metrics discussed in Section 4.2.1. Lastly, our method represents the first attempt to fully automate the process of logic checking.

While the results are encouraging, there is room for potential improvement by fine tuning the model parameters, using more advanced architectures such as Transformers [13]. Similarly, our model is validated on a small set of construction projects. Training and testing on larger datasets would help fully understand and validate the effectiveness of this approach. Finally, the presented *CC* model is targeted at creating schedules for a relatively short list of activities, and it is intended for look-ahead planning as opposed to creating an entire schedule from scratch.

4.6 Conclusions and Future Work

We presented Logic Checker (*LC*) model for automating the process of checking the logical dependencies of an input schedule as well as a formulation –Constrained Conditional (*CC*) model– that enables the use of our method to create consistent schedule from a short list of unordered activities. The method leverages deep neural networks to learn best construction practices from existing records using machine learning and natural language processing techniques and as such it is completely automated and data-driven.

Our experimental results showed that the method can identify wrong links, missing links, and or unusual relationship types. The results also showed that the *CC* model learns second degree logical dependencies across different project schedules with varying levels of details. With a 5-fold validation scheme using data from real-world commercial building construction projects form the hospitality sector, the model achieved an average Precision of 85.14%, an average Recall of 95.12%, and an average F1 score of 88.3%. Similarly, the CC model formulation showed ability to improve the global accuracy and consistency of a schedule created by the LC model from a list of unordered activities.

In these experiments, we demonstrated that a specific ⟨*Action*, *Location*, *Object*, *Responsible Party*⟩ tagging helps reduce noise and allows the model to better generalize compared to directly applying the method to the entirety of activity descriptions. We also demonstrated that while our model does not explicitly learn the root cause of different dependencies, it can implicitly generalize logic relationships and predict them correctly across various root causes of their existence.

In terms of limitations, our model only validates the logical dependencies in an input schedule,

and it is designed to complement schedule critiquing systems that accommodate additional methods to measure additional schedule quality metrics such as activity duration and lags. Similarly, our work learns to highlight the correct logic dependency between predecessor and successor activities without explicitly learning their underlying reasons. Future work will focus on highlighting those root causes and assigning degrees of flexibility and critically to input schedule relationships. Furthermore, future research should study the real-time implications of deploying novel NLP-based machine solutions that learn construction scheduling domain knowledge from existing records completely automatically and apply it to validate the logic in input schedules.

4.7 Acknowledgement

This material is in part based upon works supported by the National Science Foundation [1446765,2020227]. The support and help of Reconstruct Inc and 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.

CHAPTER 5

TRANSFORMER MACHINE LEARNING LANGUAGE MODEL FOR AUTO-ALIGNMENT OF LONG-TERM AND SHORT-TERM PLANS IN CONSTRUCTION [4]

[4]This chapter is published in *Automation in Construction* [193]:
"Fouad Amer, Yoonhwa Jung, Mani Golparvar-Fard, "Transformer machine learning language model for auto-alignment of long-term and short-term plans in construction", *Automation in Construction*, Dec 2021, 103929, ISSN 0926-5805, https://doi.org/10.1016/j.autcon.2021.103929"

5.1 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*) [23]. 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 [41]. 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 [194, 195]. 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" [196].

The look-ahead plans outline which crews and trades should be at each specific location and what they should be doing at certain times [24]. 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 [194, 197, 198], 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 [26]. 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 [22]. 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 [199]. *Bridging this gap between master schedule and look-ahead plans to get the big picture has been a constant struggle* [23].

This problem is exacerbated when progress reporting is considered [200]. The master schedule and actual information about progress reported against look-ahead plans are also disconnected. Master schedules 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. 5.1). 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.

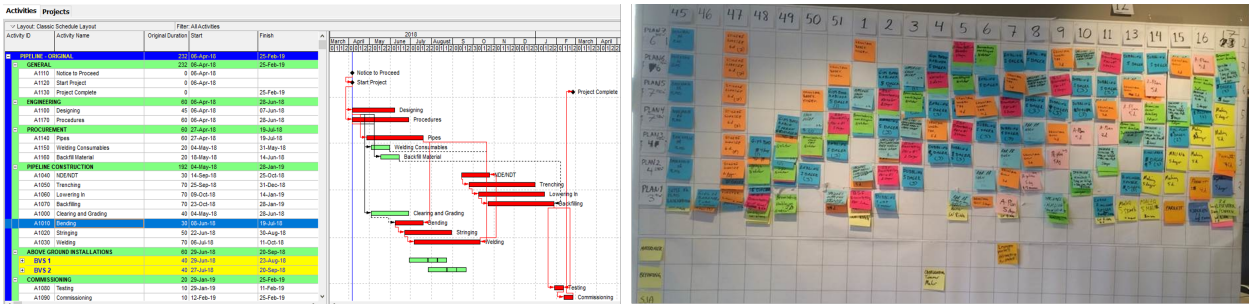


Fig. 5.1. A project manager's master schedule vs. a superintendent's look-ahead plan.

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 [200], 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 [201]. 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 [202], 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.

5.2 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 [22, 195, 203]. While there is tremendous evidence that the Last Planner System [204] improves collaboration and reduces variability in

near-term work execution [199, 205, 195, 203], 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 [23]. 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 [22]. Fig. 5.2 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.

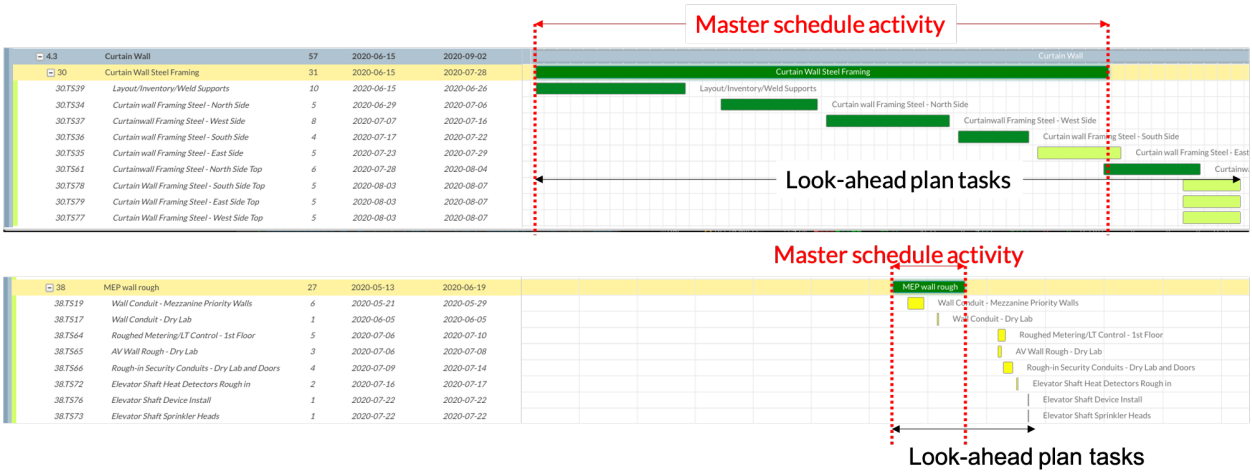


Fig. 5.2. Two examples of mis-alignments between master schedule activities and their corresponding look-ahead planning tasks (activities and tasks are shown with dark and light green colors respectively).

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 [206], keeping all plans at every level of detail in alignment proves to be a challenging and time-consuming task [23]. 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 [196]. 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 [196].

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) [22] 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 [139]. Researchers have already proposed methods for formalized representation of construction activities [122, 9, 5]. Amer et al. [34] 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 [136] 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: $\langle$*Action, Location, Object, Responsible Party*$\rangle$. Building on the ALOR method of decomposing schedule activity descriptions, Amer and Golparvar-Fard [139] formalized a new 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. [187] used Bidirectional LSTMs to learn activity sequences. Zhao et al. [188] 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 [181]. 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. [182] 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. [184] 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 [186].

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 [207, 208]. 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) [200]. After the actual site conditions are captured in 3D, the point clouds are aligned and compared to BIM to detect progress. Similarly, [209] investigated the use of RFID tags to track and control material deliveries and supply chains, and [210] 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.

5.3 Method

5.3.1 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.5.3). 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 $A = \{a_1, a_2, ..., a_j\}$, 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-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 [202], and we use it to improve matching results in a multi-stage

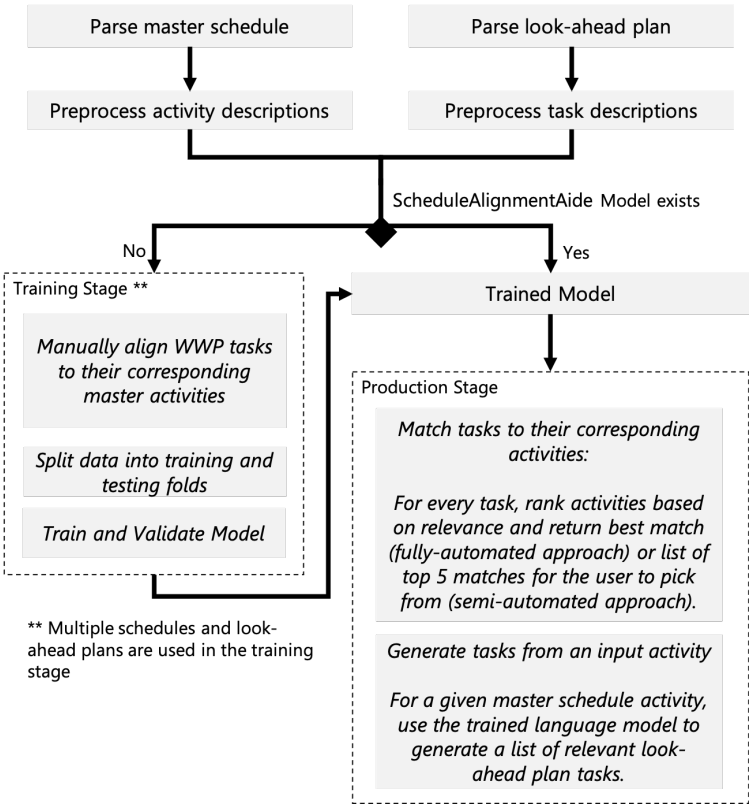


Fig. 5.3. Workflow of the proposed model at training and production times.

ranking setup [200]. 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.

5.3.2 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.

5.3.3 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 5.1 in Section 5.4.1). Second, we describe activities and tasks using FastText embeddings [158] trained on data from 32 construction schedules. As described in Amer and Golparvar-Fard [139], 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. 5.4. Third, Cosine similarity in Eq. 5.1 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

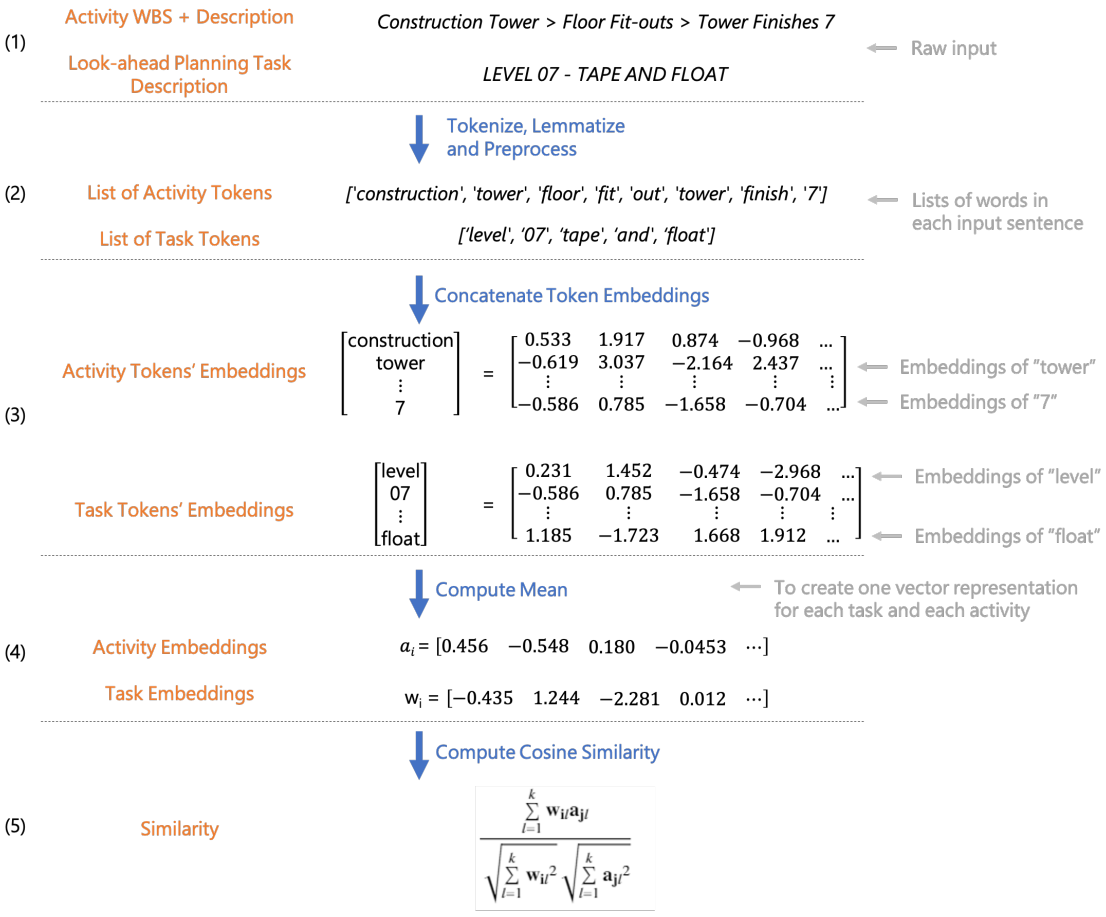


Fig. 5.4. Distance-based similarity computations between activity and task descriptions.

1 where -1 indicates that the task and activity are too distant (opposite), while 1 indicates that they are identical.

$$\text{Similarity}(\mathbf{w_i}, \mathbf{a_j}) = \frac{\mathbf{w_i} \cdot \mathbf{a_j}}{\|\mathbf{w_i}\|\|\mathbf{a_j}\|} = \frac{\sum_{l=1}^{d_k} \mathbf{w_{i}}_l \mathbf{a_{j}}_l}{\sqrt{\sum_{l=1}^{d_k} {\mathbf{w_{i}}_l}^2} \sqrt{\sum_{l=1}^{d_k} {\mathbf{a_{j}}_l}^2}} \tag{5.1}$$

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 [139].

5.3.4 Location Extraction and Matching

Locations play a significant role in construction planing, and they are identified as one of the constraining flows of construction [95]. 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 [136]. 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. 5.2):

$$lev(w_{loc}, a_{loc}) = \begin{cases} |w_{loc}| & \text{if } |a_{loc}| = 0 \\ |a_{loc}| & \text{if } |w_{loc}| = 0 \\ lev(tail(w_{loc}), tail(a_{loc})) & \text{if } w_{loc}[0] = a_{loc}[0] \\ 1 + min \begin{cases} lev(tail(w_{loc}), a_{loc}) \\ lev(w_{loc}, tail(a_{loc})) \\ lev(tail(w_{loc}), tail(a_{loc})) \end{cases} & \text{otherwise} \end{cases} \tag{5.2}$$

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.

5.3.5 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 [211], because transformers have shown superior performance on ranking problems when compared to other approaches [200]. 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 [202], which is pretrained over 40GB of Internet text. Our model can be defined as a mapping function "Matcher" (M) that maps all tuples of $\langle$ *task, activity* $\rangle$ into the binary space {0,1}:

$$M : \mathbb{W} \times \mathbb{A} \rightarrow \{0, 1\} \tag{5.3}$$

$$l_{ij} = M(w_i, a_j) \text{ s.t. } l_{ij} \in \{0, 1\} \tag{5.4}$$

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:

$$M : \mathbb{V}^2 \rightarrow \{0, 1\} \tag{5.5}$$

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:

Language Modeling

Language models –specifically forward language models– calculate the probability of a sentence (represented as a list of tokens: $t_1, t_2, ..., t_{k_N}$) by computing the probability of the next token (t_k) given all previously seen tokens ($t_1, t_2, ..., t_{k-1}$) as shown in Eq. 5.6 [212]. 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.

$$P(t_1, t_2, ..., t_N) = \prod_{k=1}^{N} P(t_k | t_1, t_2, ..., t_{k-1}) \tag{5.6}$$

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 [213]. 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 [214].

The second generation of language models is neural. These models use context-independent word embeddings –such as Word2Vec [215]– 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 [214]. 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 [216]. 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



Fig. 5.5. 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.

parsing longer sentences.

The Transformer architecture [13] 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 [217, 218]. 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 [202]. 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.

Multi-Head Attention and Transformer Decoder Blocks

GPT-2 is composed of 12 layers of Transformer Decoder blocks [211]. 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 5.1 (see Section 5.4.1), 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. 5.5. 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. 5.6, each Decoder block applies a multi-headed self-attention mechanism followed by a fully connected layer to create a distribution over the target vocabulary:

$$h^0 = LayerNorm(XW_e + W_p) \tag{5.7}$$

$$h^{(i)} = block(h^{(i-1)}) \forall i \in [1, n = 12] \tag{5.8}$$

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:

$$h^{(i)} = LayerNorm\Big(Attention(Q, K, V) + Residual(h^{(i-1)})\Big)W_D \tag{5.9}$$

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. 5.7 for an illustration of the performed computations) and is defined as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5.10}$$

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:

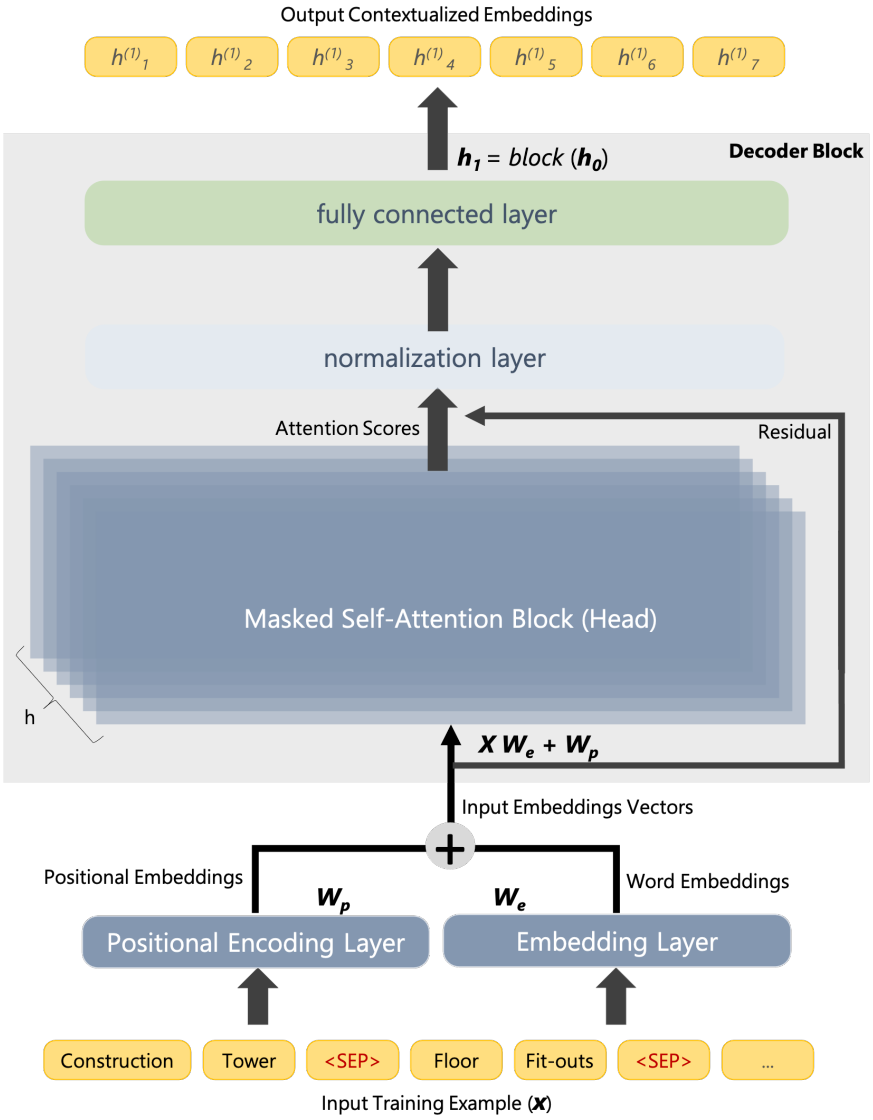


Fig. 5.6. The input, output, and architecture of the first decoder block in the GPT-2 model.

$$Q = h^{(i-1)}W_Q \tag{5.11}$$

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:

$$K = h^{(i-1)}W_K \tag{5.12}$$

This formulation of "*Queries*" and "*Keys*" enables the computation of self attention scores by simply computing QK^T as shown in Eq. 6.5. Lastly and similar to "*Queries*", "*Values*" is a transformed version of $h^{(i-1)}$ (Eq. 6.8 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. 6.5.

$$V = h^{(i-1)}W_V \tag{5.13}$$

As described by Fig. 5.7, 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. 5.7 demonstrate how the model is unidirectional. The output embeddings of block *i* are then fed into block *i+1* as shown by Eq. 6.3.

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. 5.8). The LM_{head} consists of a fully connected layer followed by a *Softmax* layer such that:

$$LM_{head} : Z_{lm} = Softmax(h_{12}W_{LM}) \tag{5.14}$$

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,...,*i*-1]. This is used to teach the model to generate look-ahead planning task descriptions given the description of a master schedule activity. As

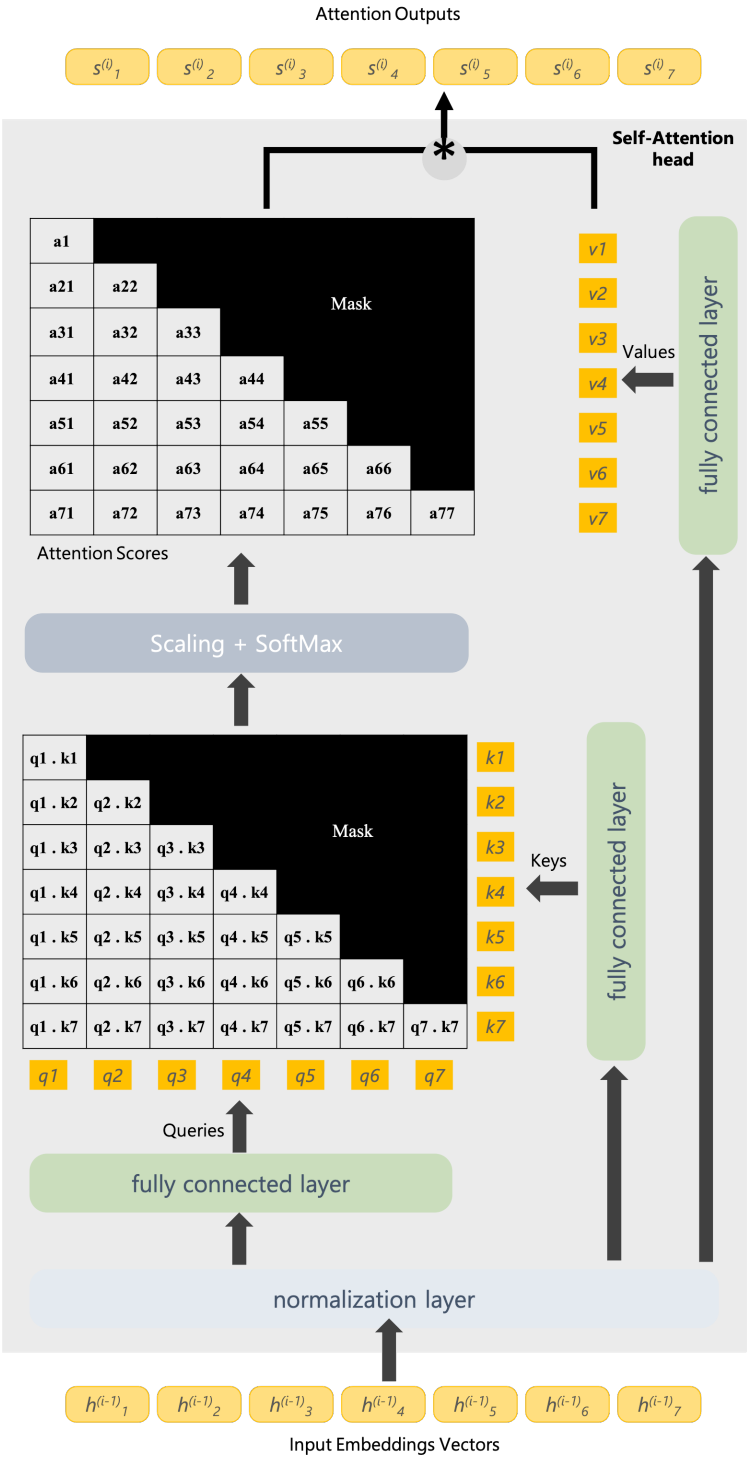


Fig. 5.7. Computations performed by a Decoder Transformer self-attention head.

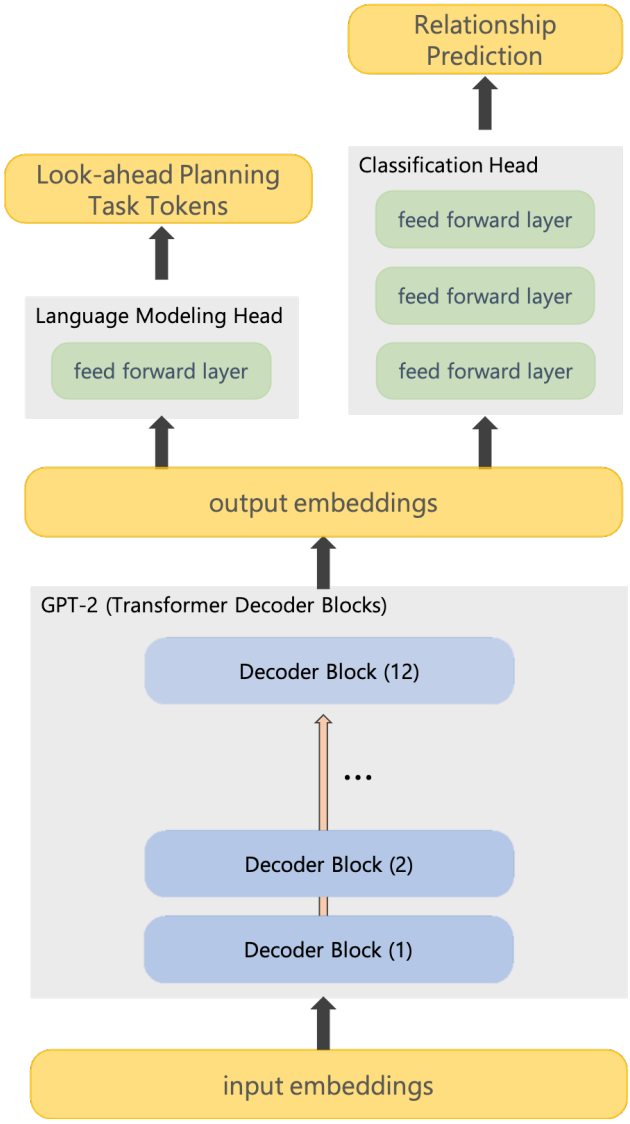


Fig. 5.8. Supervised Transformer model architecture. The GPT-2 transforms the input information of look-ahead planning tasks and master activities into the embedding space, the language modeling head learns to generate task descriptions given a master schedule activity, and the classification head learns to predict the relationship status between the task and the activity.

discussed later in Section 5.4.1, 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. 5.3. It is composed of three fully connected layers validated empirically, and it employs *LeakyRelu* activation functions followed by a *Softmax* layer as follows:

$$Z_{c_1} = LeakyRelu\big(h_{12}W_{c_1}\big) \tag{5.15}$$

$$Z_{c_2} = LeakyRelu\big(Z_{c_1}W_{c_2}\big) \tag{5.16}$$

$$Z_{c_3} = LeakyRelu\big(Z_{c_2}W_{c_3}\big) \tag{5.17}$$

$$Z_c = Softmax\big(Z_{c_3}\big) \tag{5.18}$$

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:

$$LeakyRelu\big(x\big) = max\big(0, x\big) - 0.01 \times min\big(0, x\big) \tag{5.19}$$

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. 5.18) 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 [200]. 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.

5.3.6 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.

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

Table 5.1. Examples from the data showing matches between master schedule activities and look-ahead planning tasks where an activity can be associated with multiple tasks.

5.4 Experimental Setup, Results and Discussion

5.4.1 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, 2,131 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 5.1 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 4,262, 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 5.3, 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.

5.4.2 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 5.2, 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.

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

Table 5.3 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 the ground truth is negative but the prediction is positive.

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

5.4.3 Location-Based Matching Results

Table 5.4 shows examples of the performance of ALOR tagging [136] 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.

Look-ahead Planning Task	Master Activity
level 32 vertical work	superstructure l 8 to l 36 concrete level 32
	superstructure l 8 to l 36 l 32 reshoring pour deck and column
	superstructure l 8 to l 36 l 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 5.4. Examples of location extraction and matching between look-ahead planning tasks and master schedule activities.

5.4.4 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 5.4.1. A discussion on the reasons for considering a semi-automated approach is further discussed in Section 5.5.

Table 5.5 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 5.6 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 5.2 where the f1 Score on predicting a negative relationship is 90.2%.

As shown in Tables 5.5 and 5.6, 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

	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

Table 5.5. 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.

on one of the models (the model tested on validation fold #1) and we show the results in Figures 5.9 and 5.10 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 5.7 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 milliseconds per look-ahead planning task per activity. In other words, for a look-ahead plan of 50 tasks and a master schedule of 2,000 activities, the required overall time is 10.8 seconds.

	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

Table 5.6. 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.

	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

Table 5.7. Average time in milliseconds required to retrieve the master activity match for a single look-ahead planning task.

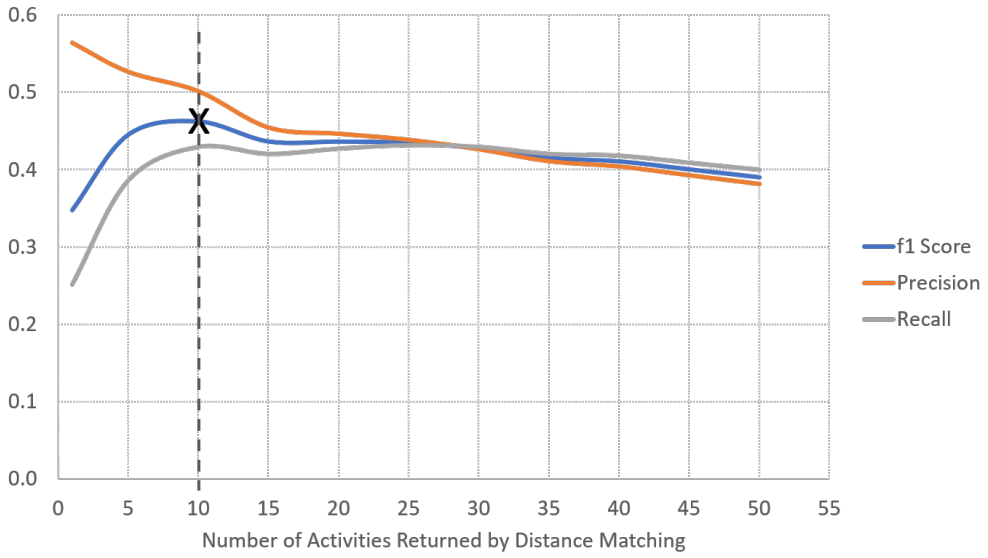


Fig. 5.9. Precision, Recall, and f1 Score of the "Distance + Transformer" model on identifying the most relevant master activity to an input look-ahead planning task based on the number of candidates generated by the distance-based matching model.

Generation of tasks from activities

As mentioned in Section 5.3, 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. 5.14, 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 pre-defined 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 [219] where k is empirically set to 25 or (2) the list of words whose sum of probabilities is equal to a p-value [220] 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 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

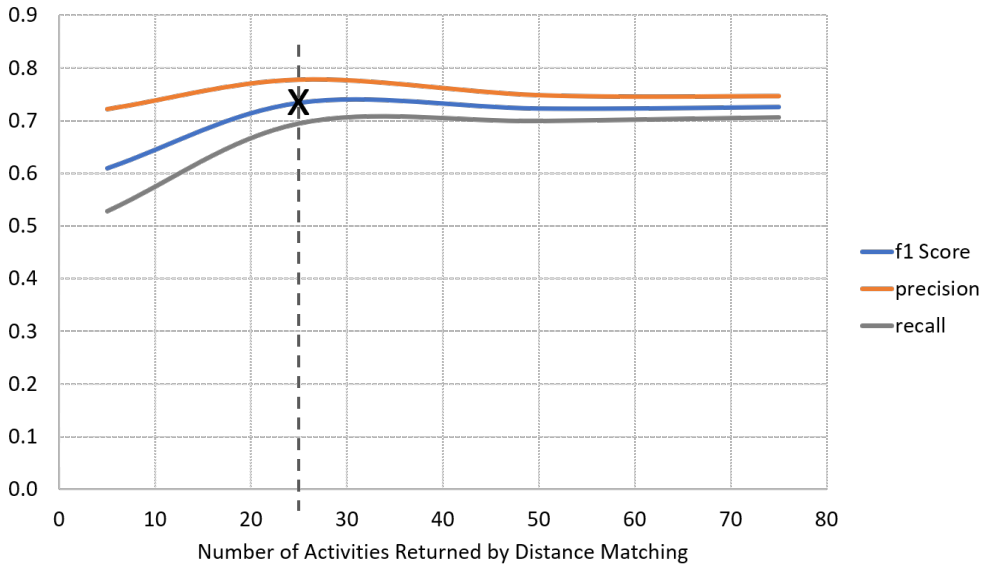


Fig. 5.10. Precision, Recall, and f1 Score of the "Distance + Transformer" model on returning a list of five master activities which contain the most relevant activity to an input look-ahead planning task based on the number of candidates generated by the distance-based matching model.

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 future research efforts. Table 5.10 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.

5.5 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 5.8 shows

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

Table 5.8. 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.

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 5.9 (Examples 1,2, and 3) show cases where the similarity in describing activity and tasks yielded correct model predictions. On the other hand, Table 5.9 (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*" (26^{th} 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.

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

Table 5.9. Qualitative examples of the predictions made by the distance-based model.

By checking the results of the "Location + Distance" model in Table 5.5, 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 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 [200]. 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 fulfill 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 5.6) to the results of the fully-automated approach (Table 5.5), 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 5.10 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 [202] 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 5.1 and 5.8) 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

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	√

Table 5.10. 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.

minimized as we train the model on larger datasets. Similarly, one other inherent limitation of data-driven approaches is the unpredictability of the nature 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.

5.6 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.

5.7 Acknowledgement

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 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.

CHAPTER 6

CONSTRUCTION SCHEDULE AUGMENTATION WITH IMPLICIT DEPENDENCY CONSTRAINTS AND AUTOMATED GENERATION OF LOOKAHEAD PLAN REVISIONS

6.1 Introduction

Construction planners often need to modify the sequencing of activities to meet contractual milestones or to disentangle process bottlenecks [10]. Nonetheless, re-sequencing activities requires (1) reasoning about the nature and flexibility of the implicit rationale of the constraints that govern the logical dependencies among them and (2) access to different sequencing alternatives. Both conditions are not available in traditional Critical Path Method (CPM) schedules. For instance, CPM schedules only model the explicit precedence relationship, i.e. Finish-to-Start (FS), Start-to-Start (SS), Start-to-Finish (SF) and Finish-to-Start (FS), and they only describe activities as critical or non-critical from a time and float perspective.

Automating the creation of revised looahead plans can potentially decrease project durations, introduce cuts in project costs, and improve planning productivity [34, 35]. To do so, an Artificial Intelligence (AI) agent needs to parse an input schedule, build a semantic representation of the implicit dependencies between its predecessor and successor activities, highlight activities that should and can be revised based on the delayed activities, and use construction sequencing knowledge to suggest an alternative schedule that minimizes the overall delay. To enable the rapid generation of revised schedules, Koo et al. [10] formalized the “role", “flexibility" and “status" of the relationships between predecessor and successor activities. To identify those attributes, Koo et al. assumed the input schedule is already augmented with dependency “constraint” information where each predecessor-successor relationship is labeled with the rationale of the constraint that demands it. For example, the relationship between “place columns” and “place slab” will be assigned a “Supported by” constraint indicating that the dependency between those two activities results from the slab’s need to be supported by the columns. Koo et al. then define the role, flexibility, and status

of the relationships based on the predefined and manually assigned constraints while enabling the planner to alter the assignments based on the project's specific conditions. Building on the work of Koo et al. [10], the first contribution of this paper is a deep learning-based Natural Language Processing (NLP) method for identifying the constraints governing the logical dependencies of an input construction schedule completely automatically without the need for any human input.

The automatic generation of construction schedules requires a formalization of construction sequencing knowledge that enables an AI to reason about the sequencing of activities and their relationships. The planning knowledge needs to be transferred from human planners to AI engines using representations that are interpretable by a computer. Traditionally, construction planning knowledge was modeled using hard-coded construction method templates [60] or fragnets [72]. Recently, Amer & Golparvar-Fard [139] used NLP tecnhiques and deep learning to learn sequencing knowledge from existing records. Simlarly, Amer et al. [167] presented a model that can schedule an unordered list of construction activities after learning sequencing knowledge from existing real-world construction schedules. Their model uses sequential machine learning to learn the explicit logical dependency between two input activities and applies a Constrained Conditional (CC) model [190] to generate a consistent schedule. Building on [167], the second contribution of this paper is a method for the automated generation of revised lookahead plans. We use the Logic Checker (LC) model developed and trained in [167], and we update the CC model to minimize schedule duration while also maximizing its consistency.

In the following section, we briefly review the literature related to (1) the formalization of construction sequencing rational, (2) the automated generation of construction schedules, and (3) the applications of NLP techniques in construction research.

6.2 Related Work

6.2.1 Formalization of Construction Sequencing Dependencies

Echeverry et al. [7] were the first to formalize construction sequencing dependencies. They clustered the "factors" governing sequencing constraints into multiple categories, namely *Physi-*

cal Relationships, *Trade Interactions*, *Code Regulations*, and *Path Interference*. They also defined different "types" of constraints within different factor categories. For instance, the physical relationship category included constraints such as *Supported By*, *Covered By*, *Embedded In* (contributing vs. noncontributing to structural function), *Relative Distance to Support*, *Relative Distance to Access*, and *Weather Protected By*. For every type of constraint, they identified whether the sequencing introduced by this constraint is modifiable (*Flexible)* of rigid (*Inflexible*) given the the construction methods put in place.

Building the the seminal work of Echeverry et al. [7], Koo et al. [10] refined the formalization of sequencing dependencies and introduced "role" and "status". Their goal was to allow construction planners to explicitly describe the rationale of their sequencing in CPM schedules to enable the re-sequencing of activities and the generation of revised look-ahead plans consistently and rapidly. They defined activities roles to be *Enabling* or *Impeding* and their statuses with respect to a target activity to be *Driving* or *Nondriving*. An activity is *Enabling* when it is needed to enable the successor activity to take place, and it is *Impeding* when it prohibits the execution of the successor activity. For example, they specified the *Supported By* to be an *Enabling* constraint and *Damaged By* to be *Impeding* constraint. Table 6.1 shows the different factors, types, roles, and flexibilities of different constraints as defined by Koo et al. [10].

Dong et al. [72] focused on the finishing and closeout phases of construction projects and introduced *Blocking* and *Zone* constraints. A *Blocking* constraint is a specific case of an *Impeding* constraint where the predecessor activity prohibits the work of the successor activity due to the spatial blocking of access such as blocking a hallway or a loading area. Similarly, a *Zone* constraint describes activities that need to be executed concurrently due to spatial reasons such as painting multiple rooms in the same construction area.

6.2.2 Activity Re-sequencing and Automated Generation of Construction Schedules

The literature of automated construction planning is very rich [34, 37]. The developed systems and methods can be split into three main categories: Knowledge-based Expert Systems (KBS), Case-based Reasoning (CBR) Systems, and Model-based Systems. KBSs refer to rule-based

Table 6.1. Factor, Type, Constraint, Role, and Flexibility of construction sequencing dependencies. Recreated based on Koo et al. [10]

Factor	Constraint	Role	Flexibility
Physical Component Relationships	Supported by	Enabling	Inflexible
	Connected to	Impeding	Inflexible
	Covered by	Enabling	Inflexible
	Enclosed by	Impeding	Flexible
	Closer to	Enabling	Flexible
	Protected by	Enabling	Flexible
Trade Interactions	Workspace	Impeding	Flexible
	Resource	Impeding	Flexible
	Damaged by	Impeding	Flexible
	Serviced by	Enabling	Inflexible
	Obstructed by	Impeding	Inflexible
Code Regulations	Safety	Impeding	Inflexible
	Inspection	Impeding	Inflexible
	Testing	Impeding	Inflexible

reasoning and decision-making methods that are built on top of artificial intelligence algorithms to solve complex problems such as [221, 78, 5, 92] among others. In these systems, knowledge is stored using exhaustive listings of if-then rules that define semantic objects and the relationships that govern them. Despite their significant contributions, these systems faced a barrier in their adoption: They were unable to automatically capture and retain knowledge learned from deploying schedules to new projects and a human planner/programmer was always needed to manually intervene and enhance the performance of such systems and grow their knowledge-bases manually [34]. To address the problem of capturing and retaining knowledge learned from deploying schedules to new projects, researchers explored the use of CBR systems to manually grow the knowledge bases of their KBSs. CBR systems model construction planning knowledge by case where each case is a records of a planning problem along with its recommended solution. When solving a new planning challenge, CBR systems assess the similarity of the new case to all the cases stored in the knowledge base. If the new case is found to be similar to an existing case, the solution of the exiting case is adapted to the new problem and returned to the planner. If approved by the planner, the CBR system retains the problem-solution tuple in its database as a new case [39]. Although CBR systems offered an early attempt to learn from planning examples, building a scalable and effective CBR system proved to be challenging particularly due to the complexity involved in (1) designing features and similarity metrics that would accurately describe planning problems as well

as (2) forging an adaptation algorithm that modifies the old case to fit the new problem [34].

Furthermore, researchers leveraged the information available in product models, such as Computer Assisted Design (CAD) models and Building Information Models (BIM), to reason about the scope of work, the relationships between physical project components, and –with the help of a knowledge base– the sequencing logic among activities [5, 11, 46]. These object-oriented automated planning systems overcome some of the shortcomings of knowledge-based systems and enabled the systematic development of scheduling alternatives based on their understanding of the project site layout and geometry. Nevertheless, the developed systems have somewhat struggled with handling inter and intra-class variability in inferring construction methods and component-specific processes from product models. This is because construction methods were manually defined using rigid work templates. For example, MOCA [127] manually modeled construction methods in the form of process templates and applied these models to attributes queried from physical project components to automatically create and sequence work activities. The rigidity of these templates and the need for manually creating them has impacted the scalability of using such systems.

6.2.3 The Applications of Natural Language Processing Techniques in Construction

NLP has been used widely in the construction research domain [181]. For information acquisition and retrieval, Zhou et al. leveraged NLP to check the conformity of building designs to the contracts' environmental requirements [184]. Lin et al. used part-of-speech (POS) tagging to facilitate the process of Requests for Information (RFIs) submittals by identifying and mapping keywords from RFIs to entities and attributes in the Industry Foundation Class (IFC) [186]. Similarly, Zhang et al. extracted syntactic and semantic features from regulatory documents and BIM using NLP techniques [182].

In addition to information retrieval, NLP techniques were also deployed in the construction planning domain to automatically learn planning knowledge from existing construction schedules. For instance, Amer and Golparvar-Fard [136] used NLP to retrieve the ⟨*Action, Location, Object, Responsible Party*⟩ (ALOR) constituents of activities following a Part-of-Activity (POA) tagging

pipeline and using activity descriptions only. Moreover, mer and Golparvar-Fard [139] built on the ALOR schema and used Long Short-Term Memory Recurrent Neural Networks (LSTMs) to formalize construction planning and sequencing knowledge and learn it from existing records. Similarly, Alikhani et al. [187] modeled activity sequences using Bidirectional LSTMs, and Zhao et al. [188] used *n*-gram models, POS tagging, and lookup tables to extract logical dependencies and construction methods from project schedules. The NLP-based research we present in this paper complements previous efforts by introducing a new method for identifying the implicit logical dependency constraints of an input schedule. The discerning of dependency constraints enables the identification of relationships flexibility and provides a basis for revising lookahead plans. The next section formulates the problem and describes the method in detail.

6.3 Method

In regular CPM schedules, the constraints that create the dependencies between activities are not explicitly available. Instead, they live in the minds of the planners who originally created the schedule. For example, planners create multiple activities to plan the execution of a painting job. They will often create an activity for applying the prime coat and another for applying the final coat, and they will link these two activities by a FS relationship. However, the reason for why this relationship exists in the schedule as a FS dependency –the fact that the prime coat is *Covered By* the final coat– is omitted because (1) it is time consuming to create for every relationship, (2) most –if not all– widely used commercial planning platforms do not require it, and (3) it is not readily available/formalized in the mind of the planner. As best explained by De La Garza [2], planners often know more than what they can explain, and their know-how proves hard to transfer to junior planners [34]. All what is documented in the schedule is a FS link between the two activities. This becomes problematic when there is need for rescheduling. When an activity is expected to be delayed, its anticipated delay propagates through its critical successors and impacts their expected end dates (see Fig.6.1). If the implicit reasons of the logical dependencies in the schedule which cause a successor to be critical are not known to the planner, being a human or an AI agent, they will not know if they can re-organize the activities in the schedule to accommodate

for the delay. On the other hand, if the planner knows that the successor activities are critical because of a flexible constraint (such as competing resources), then they can solve it by introducing new crews or shifting activities around for example. In other words, the omitted logical constraints are essential to understanding the flexibility of the dependencies between the delayed activity and its successors and hence the available options for rescheduling.

In this paper, we offer a method to help planners reschedule activities to accommodate for unavoidable delays. To do so, we make the assumption that flexible schedule constraints can be modified when rescheduling activities while inflexible constraints need to be kept intact. Since the flexibility of logical constraints is not readily available in input schedules, we first present a new method for augmenting construction schedules with implicit dependency constraints using NLP. Once the schedule is enriched with implicit dependency information, we then leverage the knowledge of relationship flexibility to revise lookahead plans and to account for anticipated activity delays.

Formally, given an input schedule, we first identify the nature of the constraints and dependencies that govern all predecessor and successor relationships. For example, given activity *"waterproofing at tile floor"* and its successor *"ceramic tile floor"*, we first identify that *"waterproofing at tile floor"* is *Covered By "ceramic tile floor"* and that is the cause of their dependency. Then knowing that a *Covered By* constraint is *Inflexible* (Table 6.1), we learn that the relationship between these two activities is to be kept unchanged when revising the schedule. On the other hand, given activities *"paint first floor area a"* and *"paint first floor area b"*, we infer that these two activities are governed by a *Resource* dependency. Since a *Resource* dependency is a *Flexible* dependency (Table 6.1), we learn that the relationship between these two activities can be relaxed or modified when revising the schedule.

To do so, we first use NLP and machine learning to teach a model –which we call *Implicit Logic Checker* (ILC)– the implicit constraints between pairs of input activities. Second, given a delayed activity, we identify the subset of look-ahead activities which are expected to be impacted by the delay. Third, we deploy the trained model to predict the dependency constraint between all predecessor-successor pairs of the identified lookahead plan activities. Fourth, we use the Logic Checker (LC) model and the Constrained Conditional (CC) model [190] developed by Amer et al. [167] (Chapter 4) to reschedule the flexible dependencies and create a new revised lookahead plan.

Different from Amer et al. [167], the CC model is redesigned to optimize for schedule duration in addition to optimizing for the schedule's logical dependencies. In the next sections, we discuss the details of the new ILC and the modified CC models.

6.3.1 Implicit Logic Checker: Augmenting Construction Schedules with Implicit Dependency Constraints

The ILC model is an NLP-based model trained using machine learning to predict the constraints, i.e. the implicit reasons of logical dependencies, governing the relationships of predecessor and successor activities in construction schedules. The input to the ILC model is a construction schedule containing predecessor and successor relationships. Each schedule is a directed graph where the nodes are activities and the relationships are directed edges. The output of our model is a set of labels for every schedule relationship (graph edge) that describe the constraint that caused the relationship to exist. This work builds on the work developed by Koo et al. [10]. We adopt their definitions of constraints and we limit them to the set defined in Table 6.1. The set of constraints considered by our model is the set $\mathbb{C}$ defined in Eq.6.1:

$$\mathbb{C} \doteq \Big\{ \textit{Supported by, Connected to, Covered by, Enclosed by, Closer to, Protected by, Workspace, Resource, Damaged by, Serviced by, Obstructed by, Safety, Inspection, Testing} \Big\} \tag{6.1}$$

The ILC model handles each predecessor-successor pair independently. For every pair, the input of the model is composed of the WBS tag and the activity description of both the predecessor and the successor activities and the output is one or more constraints from $\mathbb{C}$ as shown in Fig.6.2.

As shown in Fig. 6.2, both predecessor and successor activities information are fed into our model as a single sentence where the WBS tag and the activity description for are separated using the special token <SEP>, and the predecessor and successor information are separated using another special token <EOS>. Both special tokens enable the model to learn the distinction between predecessor and successor information. Similarly, <CLS> is used to indicate the end of the input example where the prediction should take place.

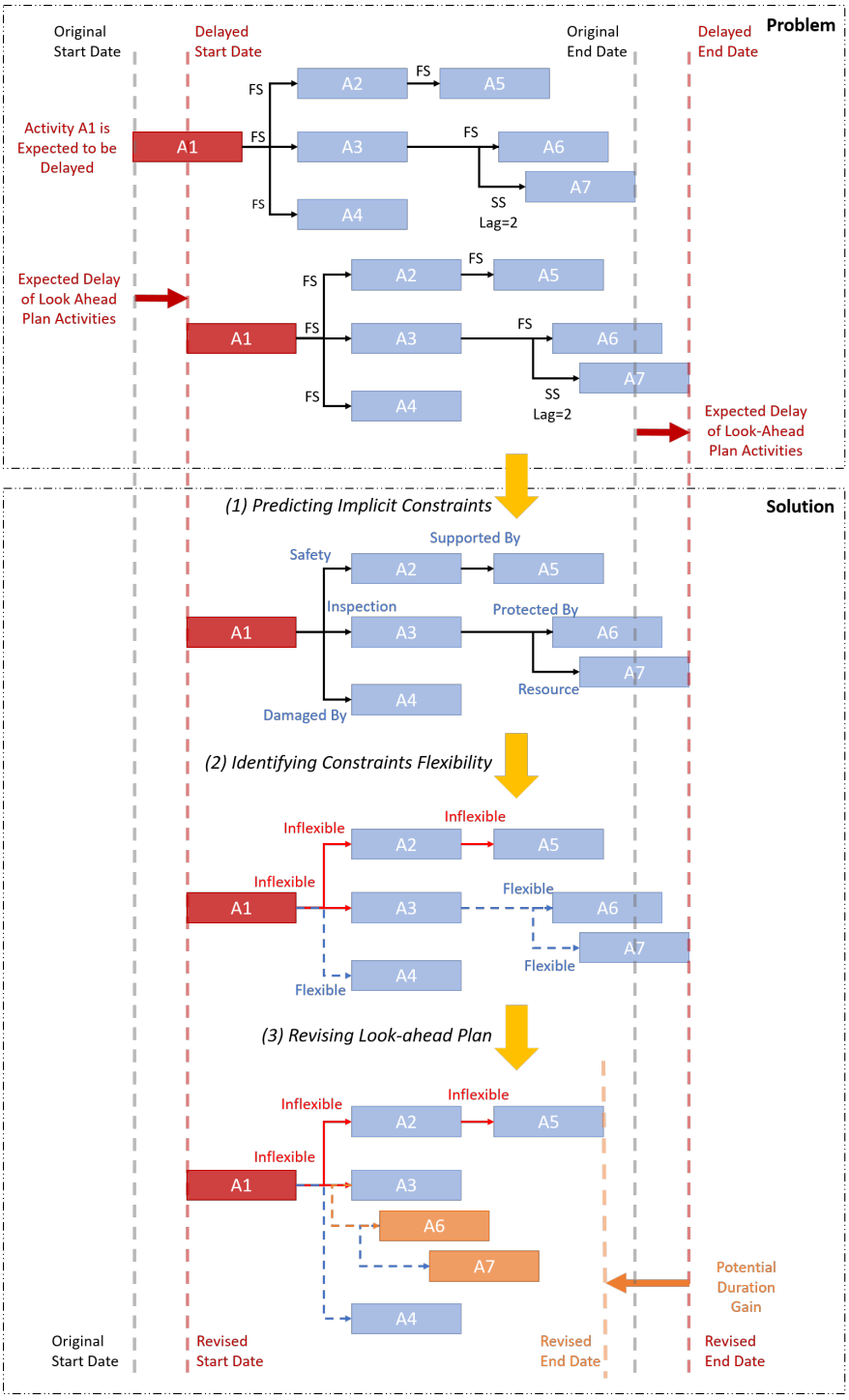


Fig. 6.1. Construction activities often get delayed, and their delay propagates through their successors in the lookahead plan (**the problem**). To mitigate some of those unavoidable delays, (1) we use the ILC model to predict the implicit dependency constraints of the lookahead plan, (2) we identify he flexibility of each relationship based on its dependency constraint, and (3) we generate a revised lookahead plan by respecting the inflexible constraints and modifying the flexible constraints using sequencing knowledge learned from previous construction records (proposed **solution**).

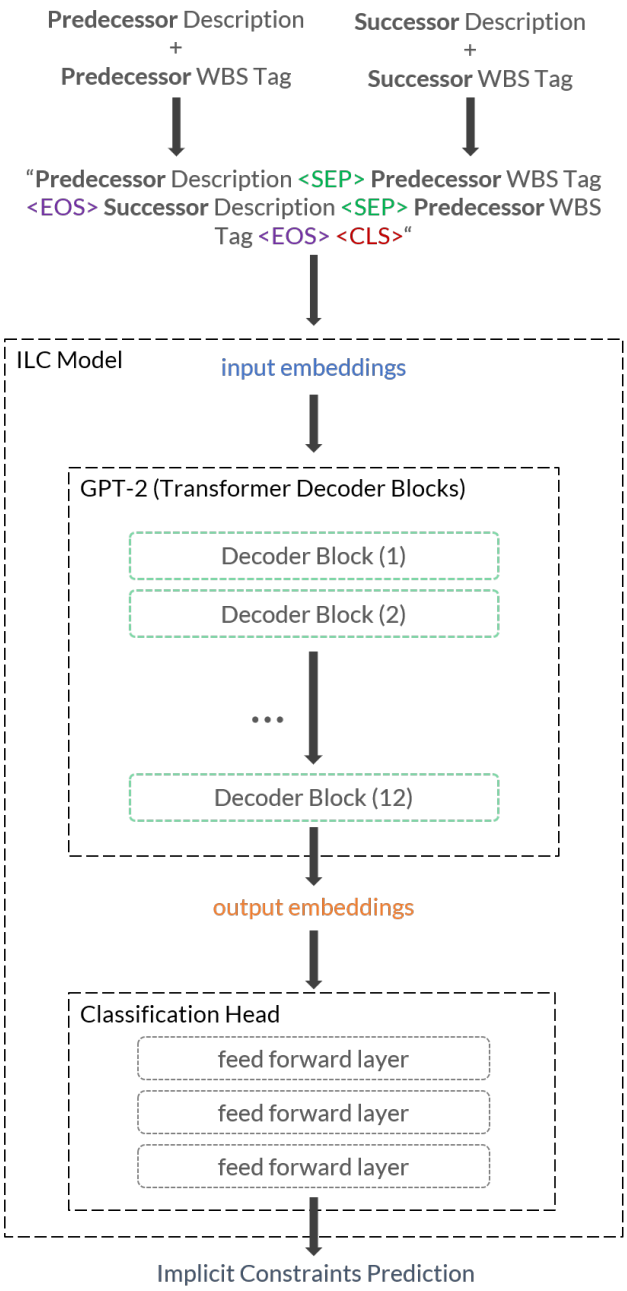


Fig. 6.2. ILC Model Architecture, Input, and Output: The model ingests the descriptions and WBS tags of a pair of predecessor and successor activities, transforms them using a series of decoder blocks and a classification head, and outputs one or more dependency constraints defined in Eq.6.1. The hyperparamters of this model are chosen empirically based on model performance, and additional hyperparameter tuning might further improve model performance.

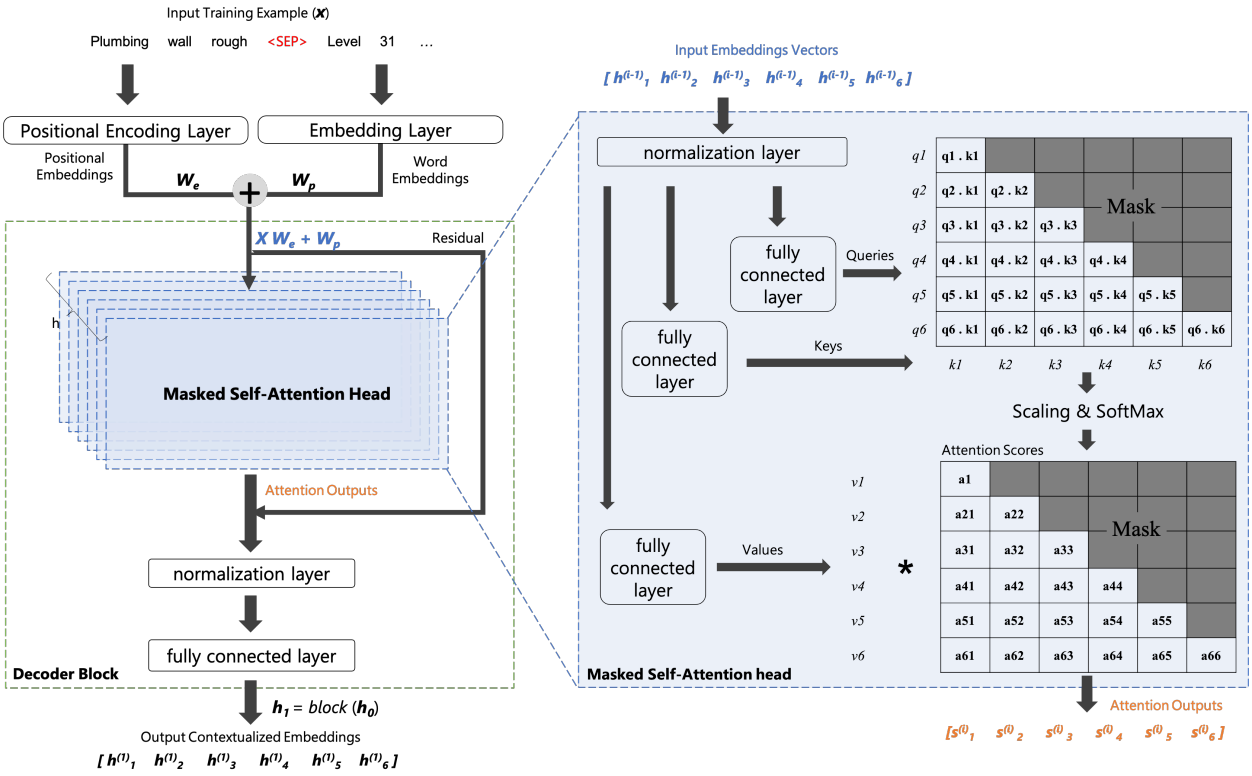


Fig. 6.3. The architecture of transformer decoder blocks and the input, output and operations performed by self attention heads.

To perform this multilabel classification, we use the state-of-the-art Decoder Transformer [211] architecture GPT-2 developed by OpenAI [202]. Different from previous neural models that rely on recurrence such as Recurrent Neural Networks (RNNs), Long-Short Term Memory RNNs (LSTM-RNNs), or Gated Neural Networks (GNNs) [214], Transformers [13] are models that rely only on "attention" mechanisms [216]. They remove the reliance on recurrence which enables them to perform better using less training data. The GPT-2 model is composed of 12 layers of Decoder blocks and is pretrained without supervision on 40GB of text data from the Internet. This pretraining enables the model to be easily reconfigured for downstream tasks such as the multilabel classification task of this study.

Before being fed to the decoder blocks, input examples are first embedded using an embedding layer and a positional encoder layer to enable the decoder blocks to learn to relative location of different words in the input example. Given X as an input example, W_e as the weight matrix of the embedding layer, W_p as the weight matrix of the positional encoding layer, the input embedding

sto be fed the decoder blocks h^0 is defined as:

$$h^0 = LayerNorm(XW_e + W_p) \tag{6.2}$$

where *LayerNorm* is a normalization function to regularize the output and ensure numerical stability when training the model.

Each decoder block than applies multi-headed self-attention mechanisms followed by normalization and fully connected layers to generate a contextualized output embeddings:

$$h^{(i)} = block(h^{(i-1)}) \forall i \in [1, n = 12] \tag{6.3}$$

where $h^{(i-1)}$ is the embeddings to decoder block i and the output of decoder block *i-1*. *block* is the mapping function –or set of computations– performed by each decoder block, and it can be defined as:

$$h^{(i)} = LayerNorm\Big(Attention(Q, K, V) + Residual(h^{(i-1)})\Big)W_D \tag{6.4}$$

where *Attention* is the mapping function performed by the masked self-attention head (Fig. 6.3), *Residual* is an input stream that allows the downstream decoder blocks to perform computations on the initial model inputs, and W_D is the matrix containing the weights of the fully connected layer of the decoder block. *Attention* is a transformation performed on input embeddings to enable the model to focus on the parts of the input sequence which are most important, and it is defined as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6.5}$$

where Q refers to "*Queries*", K to "*Keys*", and V to "*Values*" respectively. *Q, K, V* are learned transformations of the original input embeddings that enable the model to perform the attention computations, and they are defined using Equations 6.6, 6.7, and 6.8 respectively where W_Q, W_K, W_V are learnable weight matrices. See Fig. 6.3 for an illustration of the computations performed by decoder blocks and masked self-attention heads.

$$Q = h^{(i-1)}W_Q \tag{6.6}$$

$$K = h^{(i-1)} W_K \tag{6.7}$$

$$V = h^{(i-1)} W_V \tag{6.8}$$

At the end of the last decoder block computations, the output embeddings are fed into a multilabel classification head as shown in Fig. 6.2. The output of the classification head is a score for each dependency constraint defined in Eq.6.1. Every constraint that scores more than 0.5 is returned by the model as a predicted constraint, i.e. each input example (relationship between a predecessor and a successor) can have one or multiple dependency constraints. Once the constraints are assigned, Table 6.1 is used to deterministically assign a role and a flexibility to each precedence relationship in the input schedule. Koo et al. [10] argued that constraint flexibility can be project dependent, therefore we view the prediction of our model as a suggestion to the planner which can be manually altered to fit the specific project conditions. The next section discusses the data used to train and test the ILC model.

Data

The ILC model was trained on manually labeled data from eight construction projects. Two experts labeled a total of 35,332 relationships with one or more dependency constraints. Table 6.2 shows the distributions of the number of constraints per relationship in different projects: 79.6% of the dependencies can be associated with one constraint, 15.2% with two constraints, 5% with three constraints, and only 0.2% can be associated with four constraints. Fig.6.4 shows the distribution of constraints across different projects with *enclosed by* and *supported by* being the most frequent and *testing* and *safety* being the least frequent. Similarly, Fig.6.5 shows the distributions of constraints per project highlighting how these constraints can vary significantly from project to project.

To train and test the ILC model, we use a 5-fold validation scheme. We split the data into five folds and we train five different models. Each model is trained on four folds and tested on the fifth –left out– fold. We use Precision, Recall, and F1-Score to validate model performance and the results are discussed in Section 6.4. The next section presents the modified CC model and the

Table 6.2. The distribution of the number of relationships over the number of dependency constraints per relationship. For example, in project #8, 304 relationships are governed by one dependency constraint and 10 relationships are governed by two dependency constraints.

		Project							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Number of Constraints per Relationship	**1**	3024	787	1769	12525	1388	6529	1803	304
	2	349	101	325	4028	341	206	22	10
	3	65	5	23	1549	137	0	1	0
	4	0	0	0	41	0	0	0	0
	Totals	**3438**	**893**	**2117**	**18143**	**1866**	**6735**	**1826**	**314**

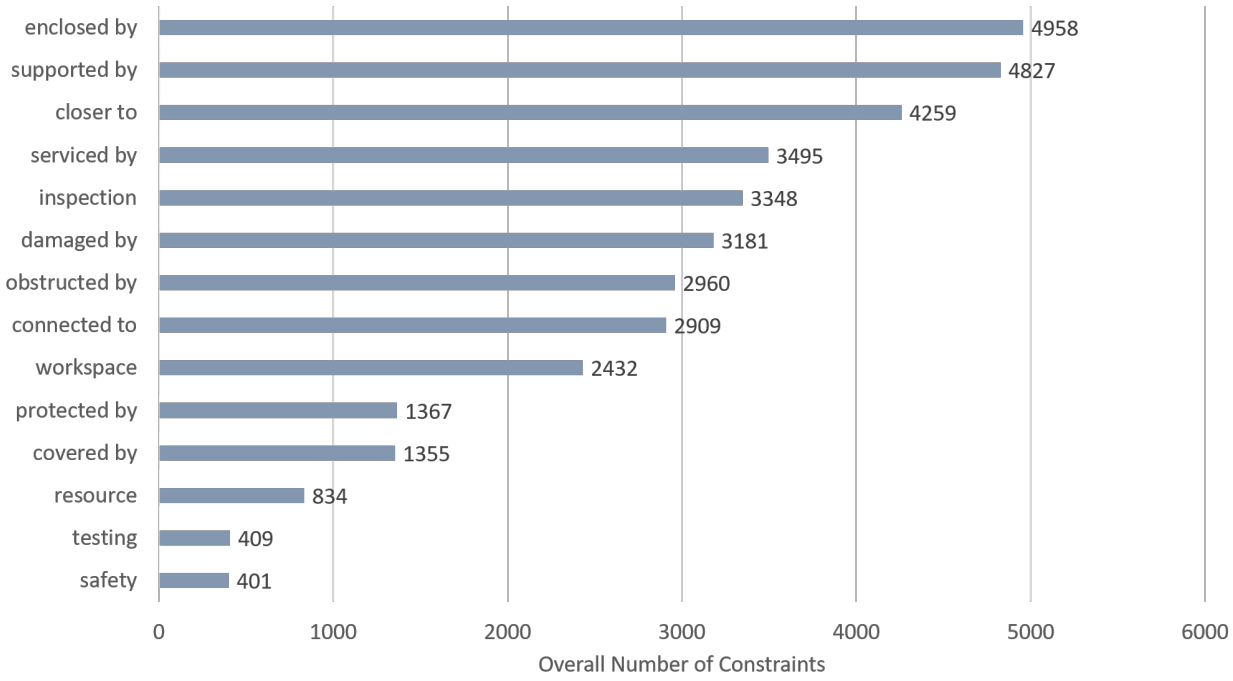


Fig. 6.4. Distribution of dependency constraints across all eight construction projects.

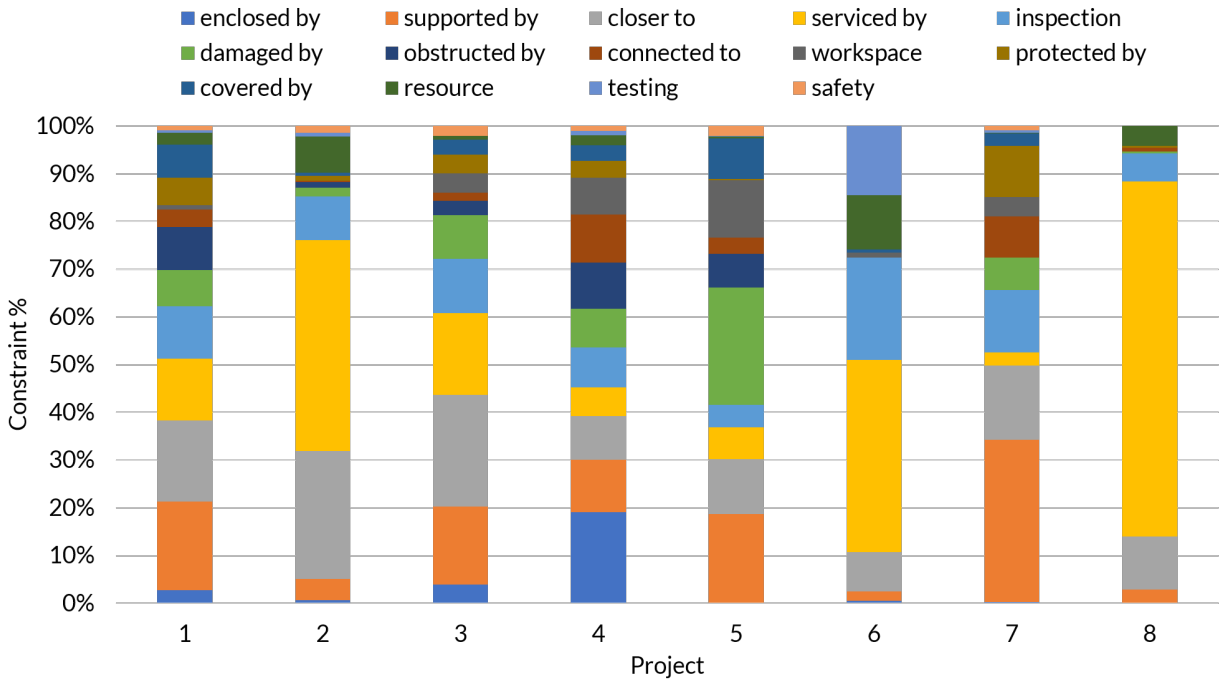


Fig. 6.5. Distributions of dependency constraints per construction project.

process of generating revised lookahead plans.

6.3.2 Generating Optimized Lookahead Plan Revisions

The LC model presented in Chapter 4 (Amer et al. [167]) maps a pair of input activities to the set of all possible explicit scheduling relationships defined by the set $\mathbb{D}$ in Eq.6.9.

$$\mathbb{D} \doteq \{FS, FS_f, SF, SF_f, SS, SS_f, FF, FF_f, NR\}. \tag{6.9}$$

The LC model parses two input activities and predicts the type and direction of the relationship between them. Using its learned construction scheduling knowledge, the LC model was combined with a CC model to create a schedule from an unordered list of input activities while optimizing the overall schedule consistency by solving Eq.6.10 using A* search [191]. $\mathbf{r}^*$ is the optimal solution vector which holds a relationship assignment for every pair on input activities. $\sum Z_{LC}(\mathbf{p}, \mathbf{r})$ is the total score of assigning the relationships predicted by the *LC* model and $d_{C_1}(\mathbf{p}, \mathbf{r})$ is the total consistency cost of that assignment.

$$\mathbf{r^*} = \text{argmax}\left(\sum_{i=1}^{n} Z_{LC}(\mathbf{p}, \mathbf{r}) - d_{C_1}(\mathbf{p}, \mathbf{r}) \right) \tag{6.10}$$

Solving Eq. 6.10 only optimizes for schedule consistency. To create optimal schedules from a time perspective, we update Eq. 6.10 to be:

$$\mathbf{r^*} = \text{argmax}\left(\sum_{i=1}^{n} Z_{LC}(\mathbf{p}, \mathbf{r}) - d_{C_1}(\mathbf{p}, \mathbf{r}) - d_{C_2}(\mathbf{p}, \mathbf{r}) \right) \tag{6.11}$$

where $d_{C_2}(\mathbf{p}, \mathbf{r})$ is the duration cost of assigning relationships $\mathbf{r}$ to activity pairs $\mathbf{p}$, i.e. the overall duration of the schedule created by assigning a specific set of relationship predictions $\mathbf{r}$ to their corresponding activity pairs $\mathbf{p}$. Formally, we define d_{C_2} to be the duration of the critical path (CP) of the schedule created by assigning $\mathbf{r}$ to $\mathbf{p}$:

$$d_{C_2}(\mathbf{p}, \mathbf{r}) = Duration\big(CP(\mathbf{p}, \mathbf{r})\big) \tag{6.12}$$

The modification introduced to the CC model by adding d_{C_2} renders the $\mathbf{r^*}$ optimal in terms of consistency and duration. And, the use of the LC model allows us to generate lookahead plans using logic learned from existing records.

The following section presents the results of testing the ILC model and validating the process of revising lookahead plans using two examples simulated from an actual construction project.

6.4 Results and Discussions

6.4.1 ILC Model Validation

As mentioned in Section 6.3.1 The ILC model was trained and tested on 35,332 manually annotated samples from eight construction projects following a 5-fold validation scheme. The detailed mean and standard deviation of Precision and Recall across five validation folds are shown in Table 6.3. Across different constraint types, the average Precision, Recall, and F1-Score are 0.96, 0.86, and 0.91 respectively.

Table 6.4 shows qualitative test examples of the predictions made by the model as well as the

Table 6.3. Mean and standard deviation of the Precision, Recall and F1-score of the ILC model for each dependency constraint across five validation folds.

	Recall		Precision	
Constraint	μ	σ	μ	σ
Serviced By	0.932	1.57%	0.943	2.22%
Safety	0.639	3.68%	0.975	2.42%
Testing	0.903	4.25%	0.974	2.14%
Workspace	0.874	3.75%	0.951	4.87%
Connected To	0.908	1.96%	0.968	0.58%
Resource	0.774	5.96%	0.960	1.08%
Protected By	0.807	2.94%	0.939	2.05%
Enclosed By	0.953	0.83%	0.976	0.45%
Supported By	0.900	3.16%	0.915	3.91%
Closer To	0.803	6.18%	0.971	1.87%
Inspection	0.966	1.44%	0.971	1.53%
Covered By	0.800	3.77%	0.956	2.82%
Damaged By	0.906	0.97%	0.975	0.67%
Obstructed By	0.842	3.85%	0.972	0.92%

ground truth constraint dependencies. For instance, the model accurately predicted that "*stair 4 tower*" is *Supported By* "*tower steel erection*". Similarly, it identified that "*ceiling drywall and tape l 19*" is is *Covered By* "*prime and 1 st coat paint l 19*". On the other hand, the model falsely predicted that "*ceramic tile floor and wall men*" is *Damaged By* "*waterproofing at tile floor*" while the true constraint dependency is *Covered By*.

6.4.2 Generating Optimized Lookahead Plan Revisions using the modified CC Model

We validate the pipeline of generating revised lookahead plans using two case studies simulated from the same input schedule. For each case study, we randomly picked an activity from the schedule and we assumed it will be delayed. The reasons why the activity is delayed are not part of the scope of this work, and we are only interested in mitigating the effects of the delay on the successor activities. As shown in Fig. 6.1, after identifying the delayed activity, we identify all the successor activities that are within a predefined lookahead planning period. In the master schedule used for validation, the durations of activities are in the range of 40 days (8 working weeks).

Table 6.4. Qualitative examples of the predictions performed by the ILC model. The "Predicted" column shows the outcome of the ILC model while the "Ground Truth" column shows the ground truth constraints.

Predecessor		Successor		Constraints	
Description	**WBS Tag**	**Description**	**WBS Tag**	**Predicted**	**Ground Truth**
structural steel erection l 4 and l 5 south	*l 4*	*structural steel detailing metal deck and shear stud l 4 and l 5 south*	*l 4*	*Connected To Serviced By*	*Connected To Serviced By*
curtainwall erection l 11	*l 11*	*curtainwall erection l 12*	*l 12*	*Supported By*	*Supported By*
tower steel erection	*parking garage structure*	*stair 4 tower*	*phase 2 tower connector*	*Supported By*	*Supported By*
mechanical riser l 7	*l 7*	*framing wet wall and mechanical room shaft l 7*	*l 7*	*Obstructed By Damaged By*	*Obstructed By*
temporary electrical and meter	*commissioning and inspection*	*mechanical system start up*	*commissioning and inspection*	*Inspection*	*Inspection*
ceiling drywall and tape l 19	*l 19*	*prime and 1 st coat paint l 19*	*l 19*	*Covered By*	*Covered By*
mechanical duct air handler and mech room l 22	*l 22*	*drywall and taping l 22*	*l 22*	*Enclosed By Covered By*	*Enclosed By Covered By*
framing and shaft 2 side l 8	*l 8*	*plumbing rough in l 8*	*l 8*	*Supported By*	*Supported By*
install ceramic tile support bldg lvl 01 bathrms rm 03	*staff restroom*	*mechanical trim out support bldg lvl 01 bathrms rm 03*	*staff restroom*	*Damaged By*	*Damaged By*
waterproofing at tile floor	*l 2*	*ceramic tile floor and wall men*	*l 2*	*Damaged By*	*Covered By*

Therefore, we look at a period of 24 weeks as our lookahead planning period in order to allow the model to see multiple successor activities and use them for rescheduling.

Case Study #1

In the first case study (Fig.6.6), activity A1 ("*tape and finish level 24*") is delayed. The successors of activity A1 that end within a period of 24 weeks are activities A2, A3, A4, and A5, therefore our lookahead plan includes those activities and their immediate successors only (A6 and A7), but we stop the search for additional successors and we do not include the successors of activities A6 and A7. Fig.6.6 shows the original Activity-on-Node (AON) network of the retrieved lookahead plan where the duration is 216 days. Following the pipeline detailed in Fig.6.1, the original lookahead plan is fed to the ILC model to identify the dependency constraints among all its predecessor-successor pairs. The model predicted that the relationships between A3 and A4, A3 and A5, and A4 and A7 are inflexible while other relationships in the lookahead plan are flexible. For instance, the model mistakenly predicted that the relationship between "*prime and paint*" and "*tape and finish*" is flexible, while these two activities are governed by a *Covered By* constraint which in inflexible. Based on the output of the ILC model, we feed a list of activities to the CC model for optimization and we anchor the relationships between the inflexible pairs of activities to stay the same in the new optimized lookahead plan. In other words, the CC model receives (1) the list of unordered activities: {A1,A2,A3,A4,A5,A6,A7} and (2) the set of anchored relationships: (A3 $\rightarrow$ A4: *FS*), (A3 $\rightarrow$ A5: *FS*), and (A4 $\rightarrow$ A7: *FS*). The CC model then searches for the relationships between all the other pairs of activities which optimize both lookahead plan duration and consistency.The results of the optimization performed by the CC model is shown in the revised AON Network (Fig.6.6). According the model, A1 and A2 can happen in parallel with a lag of 16 days, but they are both predecessors of activity A3 with a *FS* relationship. Similarly, the model predicted a *SS* relationship between A5 and A6 and removed the *FS* relationship between A4 and A6. More specifically, the model assumed that "*prime and paint*" and "*tape and finish*" can happen in parallel, and "*finish paint*" can take place in parallel with "*install cabinets and countertops*" and "*doors and hardware*". While construction professionals might argue that it's not ideal to apply prime paint in parallel to taping and finishing drywalls, this suggestion was provided by the model

due to a misprediction of the flexibility between A1 and A2. Also, it's important to note that the aim of rescheduling is to crash the schedule and optimize it to accommodate for an unavoidable delay. Hence, having to perform activities in parallel is an anticipated result of the optimization problem. Also, The outcomes of the model are intended to serve as suggestions to the site team that they can use to explore scheduling alternatives they normally might not choose. And, when deployed at a larger scale, the model serves as a tool to present scheduling alternatives based on construction logic learned from a large number of existing projects. In terms of duration, the model reduced the overall duration of the lookahead plan by 40 days and removed activity A1 from the critical path providing it with 56 days of "grace period" of delay before it impacts the original lookahead finish date.

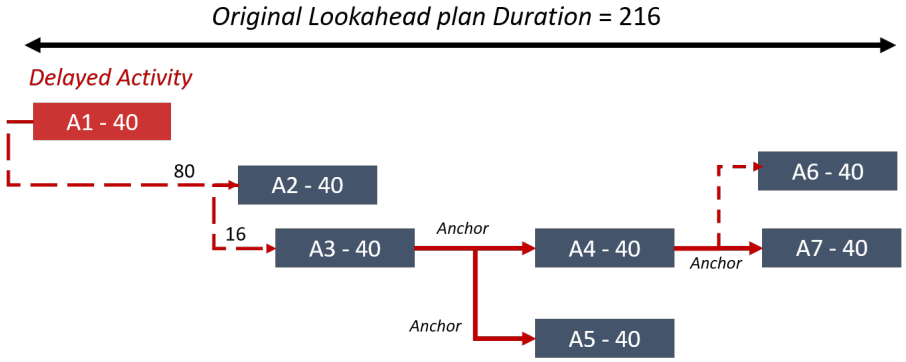
Case Study #2

In the Second case study (Fig.6.7), activity A1 ("*plumbing wall rough*") is delayed. The successors of activity A1 that are included in the lookahead plan of 24 weeks are activities {A2, A3, A4, A5, A6, A7, A8, A9}. The AON network of the original lookahead plan has a duration of 320 days, and it is shown in Fig. 6.7. Similar to the first case study, the lookahead plan is fed to the ILC model which identified the relationships between A6 and A7 as inflexible while all the other relationships were assumed to be flexible. Based on those assumptions, the CC model generated a revised lookahaed plan with a duration of 216 days as shown in Fig.6.7. The CC model assumed scheduled activities A1 ("*plumbing wall rough*"), A3 ("*mechanical wall rough*"), and A5 ("*electrical wall rough*") to happen in parallel with *SS* relationships instead of being conducted sequentially. And, similar to the first case study, the model consistently removed the dependency between "*tape and finish*" and "*prime and paint*". However, the ILC model mispredicted the dependency constraint between A2 and A4. This constraint is *Inspection* and the relationship (A2 $\rightarrow$ A4: *FS*) should have been inflexible as it is required for code compliance. Nonetheless, the site team can manually reschedule A2 to be a predecessor of A4 which would result in an overall lookahead plan duration of 256 days which is still 64 days shorter than the original duration.

Lookahead Plan Activities

Activity	Description	WBS Tag	Duration	Successors	Flexibility
A1 delayed	tape and finish level 24	Level 24	40	(A2, SS, 80)	Flexible
A2	prime and paint	Level 24	40	(A3, SS, 16)	Flexible
A3	install ceramic tile and flooring	Level 24	40	(A4, FS, 0) (A5, FS, 0)	Inflexible Inflexible
A4	install cabinets and countertops	Level 24	40	(A6, FS, 0) (A7, FS, 0)	Flexible Inflexible
A5	doors and hardware level 24	Level 24	40	-	-
A6	finish paint	Level 24	40	-	-
A7	install residential appliances	Level 24	40	-	-

Original Activity on Node Network



Revised Activity on Node Network

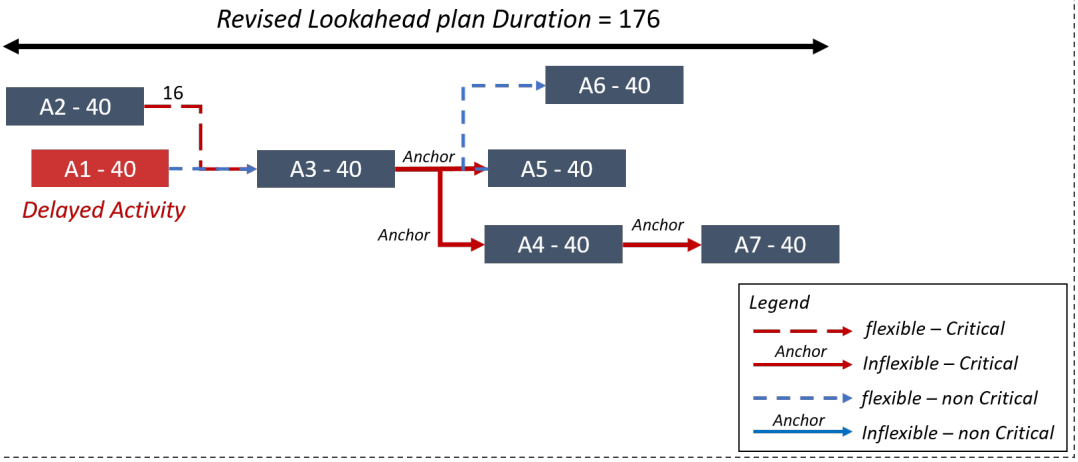


Fig. 6.6. The input lookahead plan and its original AON network as well as the revised lookahead plan for Case Study #1. For this case study, activity "*tape and finish level 24, level 24*" (A1) is delayed, and the proposed method is used to revise the lookahead plan composed of A1 and its successors that start within a period of 24 weeks of its start date.

Lookahead Plan Activities

Activity	Description	WBS Tag	Duration	Successors	Flexibility
A1 delayed	plumbing wall rough	Level 31	40	(A2, FS, 0) (A3, FS, 0)	Flexible Flexible
A2	city inspection	Level 31	40	(A4, FS, 0)	Flexible
A3	mechanical lwall rough	Level 31	40	(A2, FS, 0) (A5, FS, 0)	Flexible Flexible
A4	insulation level 31	Level 31	40	(A6, SS, 0)	Flexible
A5	electrical wall rough	Level 31	40	(A2, FS, 0)	Flexible
A6	drywall level 31	Level 31	40	(A7, SS, 24)	Inflexible
A7	tape and finish level 31	Level 31	40	(A8, SS, 80)	Flexible
A8	prime and paint	Level 31	40	(A8, SS, 16)	Flexible
A9	install ceramic tile and flooring	Level 31	40	-	-

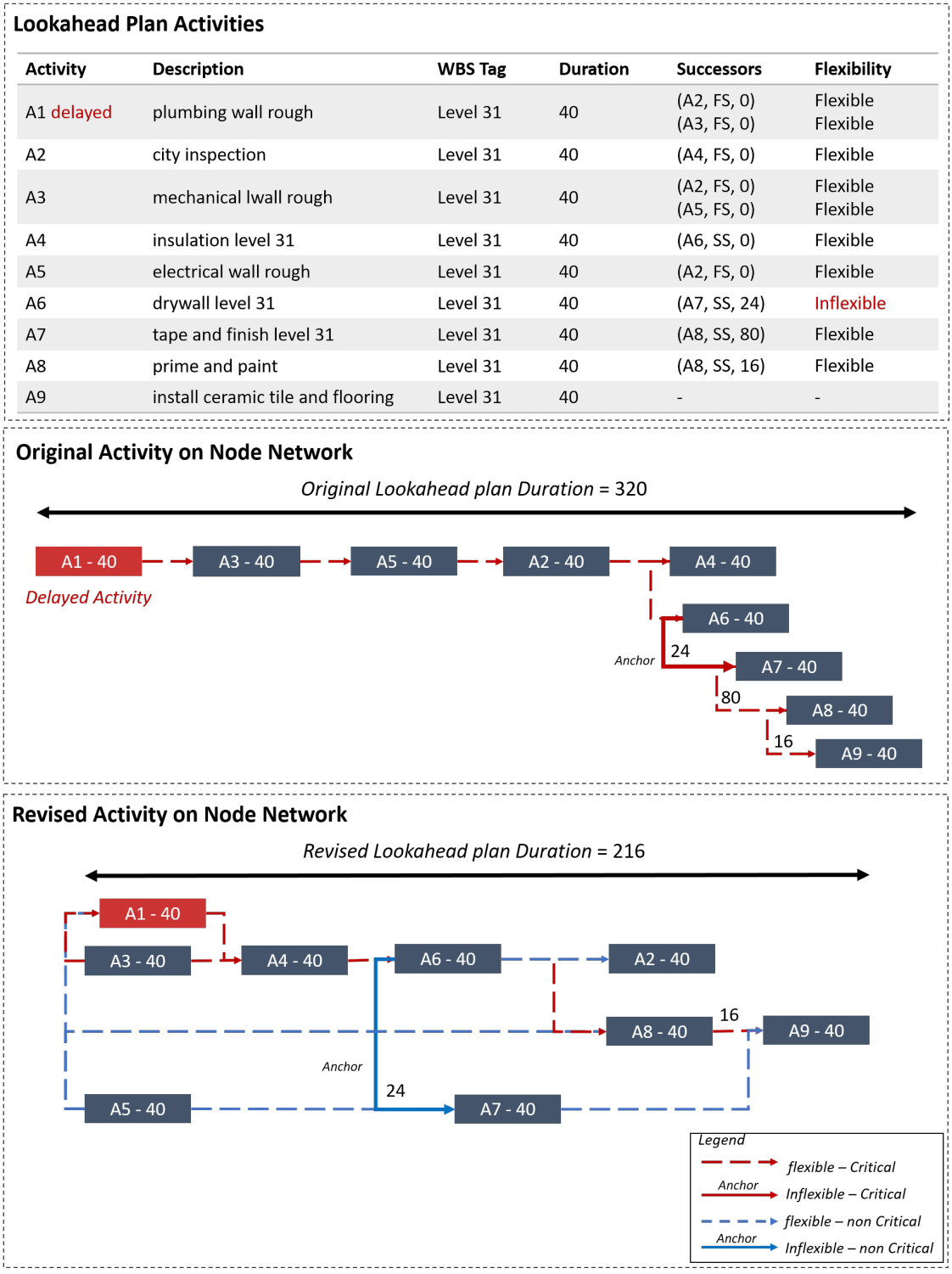


Fig. 6.7. The input lookahead plan and its original AON network as well as the revised lookahead plan for Case Study #1. For this case study, activity "*plumbing wall rough, Level 31*" (A1) is delayed, and the proposed method is used to revise the lookahead plan composed of A1 and its successors that start within a period of 24 weeks of its start date.

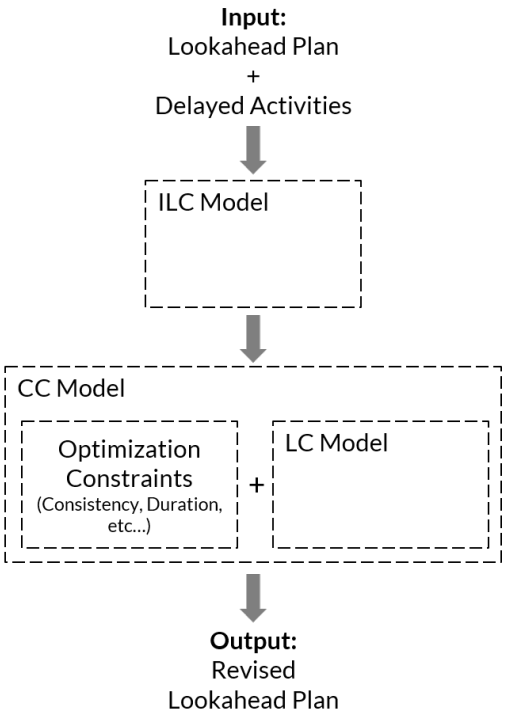


Fig. 6.8. Illustration of the components of the proposed method/pipeline and the complete decoupling between them. The proposed method ingests a delayed activity and its corresponding lookahead plan as input, and it outputs a revised optimal lookahead plan that mitigates the activity delay by reducing the overall plan duration.

Strengths and Limitations

As discussed in Section 6.3, the proposed pipeline is composed of three main components: two learning components (the LC and ILC models), and an optimization component (the CC model). By design, these components are completely decoupled from each other (see Fig.6.8). The learning components are trained and tested separately, and changing or upgrading one of them does not require any modification to the other one or the the optimization component. Additionally, the CC model enables the decoupling and the weighting of different optimization constraints. If the optimizing for the duration is more important than consistency of direct and indirect dependencies, the duration cost can be increased relative to the consistency cost. Moreover, the CC model is built on top of the LC model. And, changing the optimization criteria such as optimizing for duration, cost, work continuity, etc. does not necessitate retraining the LC model.

The optimization problem assumes that the LC model is already trained and working in pro-

duction. Furthermore, the scheduling knowledge applied for rescheduling the revised plan is that learned by the LC model. The current LC model used for this study is trained on only three projects from the hospitality sector [167]. The more data the LC model is exposed to, the better it performs and the more "reasonable" the optimized schedules created by the CC model will be. Similarly, training the ILC model on more data will lead to better predictions of constraints' flexibilities and would enable the CC model to avoid mistakes similar to those discussed in the Case Studies.

Additionally, Our proposed model is not trained to learn the lag between activities. We use the lag existing in the original schedule for the relationships in the revised lookahead plan for relationships that are maintained by the CC model. As identifying lag is essential for schedule optimization, future work can focus on expanding the LC model to predict lags in addition to dependency types. Similarly, The proposed CC model does not learn the cause of delay of the activity under consideration and it does not use that cause in the creation of the revised lookahead plan. The next section discuses the conclusions of this work as well as the recommendations for future work.

6.5 Conclusions and Future Work

We present a new machine learning-based model –which we call the Implicit Logic Checker (ILC)– for predicting the the implicit dependency constraints of an input construction schedule with an F1-Score of 91%. The predictions made by the ILC model are used to identify the flexibility of the schedule's predecessor-successor relationships. Furthermore, We design a new Constrained Conditional (CC) model that leverages the output of the ILC model to create revised lookahead plans. Those plans are optimized for duration and consistency to mitigate the effects of unavoidable activity delays. We test our pipeline on two case studies simulated from a real construction project, and we show that our model can create consistent lookahead plans with shorter durations compared to the original plans. Future work can focus on (1) deploying and validating the proposed system on an ongoing construction project, (2) expanding the constraints of the CC model to accommodate for other optimization parameters such as resources, cost, or work interruptions, (3) enhance the CC model to reason about the cause of delay to better address it in the revised lookahead plan.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

The main conclusion of this dissertation is that learning construction planning and sequencing knowledge from existing records and deploying it for solving planning problems is feasible. We proved it by building models that learn their planning knowledge from previous schedules and WWPs and deploy it to solve day to day construction planning problems such as creating lookahead plan revisions or aligning WWPs to master schedule activities. Today, there are thousands of existing construction schedules and WWPs that can be used to inform the next generation of automated construction planning engines, and those engines can be used to assist human planers revise their schedules and mitigate unavoidable delays. Existing traditional planning engines suffer from several structural scalability issues including the lack of flexibility in how their construction knowledge is stored and their dependency on manually formed and maintained work templates. In this dissertation, we validated that these bottlenecks can be overcome by learning from existing records. We showed that learning from past planning data is possible and scalable, and it can be done completely automatically with little human supervision. We showed that the learned knowledge can be stored as dynamic machine learning models, and it can be used in conjunction with optimization models to generate optimized lookahead plans. We also validated that we can teach an artificial intelligence agent to critique the logic of an input schedule based on knowledge learned from other projects. On the other hand, we showed that machine learning models can be used as an intermediate step to suggest potential solutions that can potentially facilitate the planning and control workflows of human planners and site personnel. Finally, this dissertation offered a set of models, methods, and systems to augment the knowledge of site personnel and novice planners with years of planning experience encoded in past project schedules and planning records.

Chapter 2 provides a critical literature review on the most significant body of knowledge related to automated construction planning systems. The authors examined more than 1,565 articles and synthesized results from 93 to highlight knowledge gaps that possibly hinder a wider adoption of automated construction planning solutions. This examination revealed three insights regarding why existing planning and scheduling systems cannot scale in complexity to handle data from real-world projects. The first insight is that methods for modeling and representing activities and dependencies do not enable automatically building knowledge bases using data from existing projects. Current planning data schemas and representations in the literature are rigid and do not provide semantic representation to generalize knowledge from existing projects to new projects. The second insight is that existing planning systems are not validated on complex real-life projects. They are often tested on simplified (or hand-made) planning problems where only a subset of building systems or operations is considered. The third insight is that there is a decoupling between optimization and automated planning research that causes automated planning systems to be less appealing to industry practitioners. Without an automated mechanism to generate these sequencing templates, adopting and adapting these techniques for repetitive and non-repetitive projects will remain a major challenge. Given the advancements in machine learning techniques, the authors offer a path forward towards wider adoption via automatic generation of these work templates from previous project schedules while carefully considering the granularity of project schedules, and how they can roll up and down from look-ahead plans to master-schedules.

Chapter 3 aims at enabling fast and on-demand querying of construction sequencing and planning knowledge by automatically learning the best construction practices from previous projects, and also digitizing them in a flexible and generalizable data schema. To that end, a new method is presented that models activities as vectors of embeddings built using "*Action*" and "*Object* activity constituents. The method leverages this representation to train generative sequential machine learning models to decode logical activity dependencies. The trained models which we call Dynamic Process Templates (DPT) were thoroughly tested on a dataset of 78,704 activities following three different testing schemes. The DPTs proved capable of learning construction sequencing knowledge with accuracy ranging between 76% and 98%. Our results demonstrate the robustness and flexibility in handling unseen logical dependencies and differences in natural language expressions.

Chapter 4 presents the Logic Checker (*LC*) model for automating the process of checking the logical dependencies of an input schedule as well as a formulation –Constrained Conditional (*CC*) model– that enables the use of our method to create consistent schedule from a short list of unordered activities. The method leverages deep neural networks to learn best construction practices from existing records using machine learning and natural language processing techniques and as such it is completely automated and data-driven. Our experimental results showed that the method can identify wrong links, missing links, and or unusual relationship types. The results also showed that the *CC* model learns second degree logical dependencies across different project schedules with varying levels of details. With a 5-fold validation scheme using data from real-world commercial building construction projects form the hospitality sector, the model achieved an average Precision of 85.14%, an average Recall of 95.12%, and an average F1 score of 88.3%. Similarly, the CC model formulation showed ability to improve the global accuracy and consistency of a schedule created by the LC model from a list of unordered activities. In these experiments, we demonstrated that a specific $\langle$*Action*, *Location*, *Object*, *Responsible Party*$\rangle$ tagging helps reduce noise and allows the model to better generalize compared to directly applying the method to the entirety of activity descriptions. We also demonstrated that while our model does not explicitly learn the root cause of different dependencies, it can implicitly generalize logic relationships and predict them correctly across various root causes of their existence. In terms of limitations, our model only validates the logical dependencies in an input schedule, and it is designed to complement schedule critiquing systems that accommodate additional methods to measure additional schedule quality metrics such as activity duration and lags. Similarly, our work learns to highlight the correct logic dependency between predecessor and successor activities without explicitly learning their underlying reasons. Future work will focus on highlighting those root causes and assigning degrees of flexibility and critically to input schedule relationships.

Chapter 5 presents 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. Lastly, the possibility of mapping project constraints and issues to look-ahead plans and master schedules can be investigated.

Finally, Chapter 6 We present a new machine learning-based model –which we call the Implicit Logic Checker (ILC)– for predicting the the implicit dependency constraints of an input construction schedule with an F1-Score of 91%. The predictions made by the ILC model are used to identify the flexibility of the schedule's predecessor-successor relationships. Furthermore, We design a new Constrained Conditional (CC) model that leverages the output of the ILC model to create revised lookahead plans. Those plans are optimized for duration and consistency to mitigate the effects of unavoidable activity delays. We test our pipeline on two case studies simulated from a real construction project, and we show that our model can create consistent lookahead plans with shorter durations compared to the original plans. Future work can focus on (1) deploying and validating the proposed system on an ongoing construction project, (2) expanding the constraints of the CC model to accommodate for other optimization parameters such as resources, cost, or work interruptions, (3) enhance the CC model to ingest and reason about the cause of delay to better address it in the revised lookahead plan.

7.2 Summary of Contributions

This dissertation presented a suite of machine learning and natural language processing based methods and systems for formalizing and learning construction planning and sequencing knowledge from existing planning records –master schedules and weekly work plans– and deploying the learned knowledge to:

- Generate a lists of successor activities given a sequence of predecessors and retrieve a list of predecessor activities needed to enable the execution of a sequence of successors. (Chapter 3)
- Critique and validate the quality of logic of a construction schedule by highlighting missing relationships or wrong logical dependencies. (Chapter 4)
- Create a schedule (without duration computations) from a list of unordered activities. (Chapter 4)
- Align weekly work plans to their corresponding master schedule activities following a semi-automated human-in-the-loop approach. (Chapter 5)
- Identify the implicit dependency constraints governing predecessor-successor relationships in an input schedule, and using those constraints to infer the role and flexibility of each relationship. (Chapter 6)
- Generate optimized lookahead plans in order to mitigate the effects of unexpected and unavoidable activity delays. (Chapter 6)

Moreover, we contributed **new formalization** of:

- **The Key Limitations of Existing Automated Planning Methods and Systems**. We systematically explore the literature of automated construction planning and we identify the main technical gaps hat have hindered the scaling and wide adoption of automated planning methods and systems. (Chapter 2)

- **Learnable Construction Activity Constituents**: we modify and adopt the definition of activity constituents to be the *{Action, Location, Object, Resource}* – *{ALOR}* – tuple. We validate that these constituents are learnable from existing records and build a method to learn them. (Chapter 3)

- **Construction Activity Representation**: we create a new construction activities representation using word embeddings learned from construction specific data. (Chapter 3)

- **Construction Sequencing Templates/knowledge**: we create a new method for dynamically representing construction planning and sequencing knowledge using sequential machine learning models instead of hard coded templates. (Chapter 3)

- **Construction Sequences Mining from existing records**: we contribute a new construction sequence mining method where we leverage a truncated depth first search algorithm to retrieve the sequences of activities from an input schedule graph. (Chapter 3)

- **Indirect Relationship Dependencies**: we create and define second degree (indirect) logical dependencies based on the direct dependencies explicitly available in a construction schedule and we create a conflict resolution method to identify the logical dependency between successor and predecessor activities which are six relationships a part in the schedule graph. (Chapter 4)

- **Flipped Relationship Dependencies**: We contribute and define "*flipped relationships*" to represent the direction of relationship between a successor and predecessor to enable the scheduling of an unordered list of activities. (Chapter 4)

- **Weekly Tasks to Master Activities Mappings**. We formulate the mapping of WWP tasks to master activities as a ranking problem and we develop and validate a semi-automated method to solve it. (Chapter 5)

- **Method for Revising Lookahead Plans**. We formulate, build, and validate a new automated method for generating lookahead plan revisions using learned construction planning knowledge. (Chapter 6)

We created and validated the following set of **models, systems, and methods**:

- **Part-of-Activity (POA) Tagger**: We created and validated a machine learning model built using Bidirectional LSTMs to predict the *{ALOR}* constituents of a construction activity. (Chapter 3)

- **Dynamic Process Templates (DPTs)**: We created and validated a machine learning models built using LSTMs to predict a list of successor activities given a sequence of predecessors or a list of predessessor activities given a list of successors. DPTs are supervised, but the training labels are generated automatically from the input data. (Chapter 3)

- **Logic Checker (LC) Model**: We created and validated a machine learning model built using LSTMs to predict the explicit logical dependency between two input construction activities. The LC model is supervised, but the training labels are generated automatically from the input data. (Chapter 4)

- **Constrained Conditional (CC) Model**: We built and validated an optimization model built on top of the LC model to generate a globally optimal schedule –in terms of consistency– from a list of unordered activities. (Chapter 4)

- **ScheduleAlignmentAide**: We created and validated a machine learning model built using a Transformer architecture to rank and match WWP tasks to master schedule activities. The model is intended to return the top k matching results and offering them to a human planner to finalize the matching. (Chapter 5)

- **Implicit Logic Checker (ILC) Model**: We created and validated a machine learning model built using a Transformer architecture to predict the implicit logical dependency constraint governing the relationships between predecessor and successor activities. The outputs of this model are used to identify the flexibility of different relationships and use it for revising lookahead plans. (Chapter 6)

- **Modified CC Model**: We built and validated an optimization model built on top of the LC model to generate a globally optimal lookahead plan in terms of consistency and duration. The modified CC model leverages the outcome of the ILC model to identify the flexibility of

a lookahead plan relationship, and it generates a revised lookahead plan accordingly using the knowledge learned by the LC model. (Chapter 6)

7.3 Limitations and Future Research Opportunities

This dissertation serves as a starting point in the field of learning construction planning knowledge from existing records. In addition to the discussions provided in each chapter which are specific to each proposed method, the following limitations and immediate future research opportunities can be addressed:

- **Learning Productivity Rates and Activity Durations**. In this dissertation, we did not focus on learning productivity rates or typical activity durations. This limitation is related to the type of data we worked with. This data consisted of construction schedules with preset activity durations and no information on quantities or productivity rates. Since durations are a corner stone of construction schedules, future research can focus on leveraging 4D BIM models or other quantity-enriched schedule data to teach AI agents to predict activity durations.

- **Learning from Schedule Revisions**. In this research, we focused on learning from different schedules, and we did not tailor our methods to learn from different revisions of the same schedule. This is mainly due to a imitation in our dataset where we do not have multiple revisions of the same schedules. Future research can focus on comparing revisions and learning the relationships between causes of delay and their effects on future schedules. Such models can be deployed to help planners take decisions when faced with scenarios that the models are familiar with.

- **Learning from Additional Construction Documents**. We only focused on construction schedules and weekly work plans. However, there is additional information related to planning and scheduling that can be found in other construction documents such as contracts or requests of information (RFIs). Future research can look into building new models that can learn the relationships between RFI and contract entities and planning documents to

identify patterns and provide prescriptive insights.

- **Deploying and Validating Model in Production**. Future research can also focus on deploying the proposed models on ongoing construction projects and collecting feedback from the site. Based on that feedback, researchers can update and tailor the methods to fit the needs of the site teams and provide them with the prescriptive outcomes they need. Fore instance, the developed CC model can be reconfigured to optimize for cost, resources, or work continuity in addition to duration and consistency.

- **Planning Databases and Warehouses**. hundreds –and in some cases thousands– of planning data points are being created on construction sites ever single day. Future research can work on building scalabale databases and warehouses that make this data machine learning-ready and enable the training and deployment of models at a larger scale.

- **Training on Larger Datasets**. To harvest the full potential of machine learning models, future researchers can train the proposed models on larger datasets from more construction projects.

REFERENCES

[1] H. Muñoz-Avila, K. Gupta, D. W. Aha, and D. S. Nau, "Knowledge-Based Project Planning," Tech. Rep., 2001. [Online]. Available: http://www.cs.umd.edu/{~}nau/papers/munoz-avila2001knowledge-based.pdf

[2] J. M. De La Garza, "A knowledge engineering approach to the analysis and evaluation of schedules for mid-rise construction," Ph.D. dissertation, UIUC, 1988.

[3] N. P. Garcia-Lopez, "AN ACTIVITY AND FLOW-BASED CONSTRUCTION MODEL FOR MANAGING ON-SITE WORK," Ph.D. dissertation, Stanford, 2017. [Online]. Available: http://purl.stanford.edu/jw488vx5801

[4] N. Chevallier and A. D. Russell, "Automated schedule generation," *Canadian Journal of Civil Engineering*, vol. 25, no. 6, pp. 1059–1077, 1998. [Online]. Available: http://www.nrcresearchpress.com/doi/pdf/10.1139/l98-029http://www.nrc.ca/cgi-bin/cisti/journals/rp/rp2{_}abst{_}e?cjce{_}l98-029{_}25{_}ns{_}nf{_}cjce25-98

[5] A. Darwiche, R. E. Levitt, and B. Hayes-Roth, "OARPLAN: Generating project plans by reasoning about objects, actions and resources," *Artificial Intelligence for Engineering, Design, Analysis and Manufacturing*, vol. 2, no. 03, p. 169, 1989. [Online]. Available: http://www.journals.cambridge.org/abstract{_}S0890060400000639

[6] N. Dong, D. Ge, M. Fischer, and Z. Haddad, "A genetic algorithm-based method for look-ahead scheduling in the finishing phase of construction projects," in *Advanced Engineering Informatics*, vol. 26, no. 4, 2012. [Online]. Available: http://ac.els-cdn.com/S1474034612000237/1-s2.0-S1474034612000237-main.pdf?{_}tid=6da03532-4656-11e7-b8d9-00000aacb361{&}acdnat=1496272447{_}d48438635286b2d0c7c274a042d4f3c3 pp. 737–748.

[7] D. Echeverry, C. W. Ibbs, and S. Kim, "SEQUENCING KNOWLEDGE FOR CONSTRUCTION SCHEDULING," *Construction Engineering and Management*, vol. 117, no. 1, pp. 118–130, 1991.

[8] K. K. Han, D. Cline, and M. Golparvar-Fard, "Formalized knowledge of construction sequencing for visual monitoring of work-in-progress via incomplete point clouds and low-LoD 4D BIMs," *Advanced Engineering Informatics*, vol. 29, no. 4, pp. 889–901, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.aei.2015.10.006

[9] C. Hendrickson, C. Zozaya-Gorostiza, D. Rehak, E. Baracco-Miller, and P. Lim, "An expert system for construction planning," *Journal of Computing in Civil Engineering*, vol. 1, no. 4, pp. 253–269, 1987.

[10] B. Koo, M. Fischer, and J. Kunz, "Formalization of Construction Sequencing Rationale & Classification Mechanism to Support Rapid Generation of Sequencing Alternatives," *Environmental Engineering*, no. October, 2006.

[11] R. Morkos, "Operational efficiency frontier: Visualizing, manipulating, and navigating the construction scheduling state space with precedence, discrete, and disjunctive constraints," Ph.D. dissertation, Stanford University, 2014.

[12] S. Hochreiter and J. J. Urgen Schmidhuber, "Long Short-Term Memory," Tech. Rep. 8, 1997. [Online]. Available: http://www7.informatik.tu-muenchen.de/{~}hochreithttp://www.idsia.ch/{~}juergen

[13] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," Tech. Rep., 2017. [Online]. Available: https://arxiv.org/pdf/1706.03762.pdf

[14] M. A. Bragadin, "Quality Driven Shceduling in Construction," Ph.D. dissertation, Tampere University of Technology, Finland, 2018.

[15] J. M. De La Garza and C. W. Ibbs, "Knowledge-elicitation study in construction scheduling domain," *Journal of Computing in Civil Engineering*, vol. 4, no. 2, pp. 135–153, 1990.

[16] S. F. Moosavi and O. Moselhi, "Schedule assessment and evaluation," in *Construction Research Congress 2012: Construction Challenges in a Flat World, Proceedings of the 2012 Construction Research Congress*, 2012, pp. 535–544.

[17] S. F. Moosavi, O. Moselhi, and F. Asce, "Review of Detailed Schedules in Building Construction," 2014.

[18] S. Han, J. O. Choi, and J. T. O'Connor, "Quality of Baseline Schedules: Lessons from Higher Education Capital Facility Projects," *Journal of Professional Issues in Engineering Education and Practice*, vol. 143, no. 1, p. 04016017, jan 2017. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%29EI.1943-5541.0000302

[19] R. Kenley and O. Seppänen, *Location-Based Management for Construction*, 1st ed. Routledge, sep 2010.

[20] J. T. O'connor, W. O'brien, and S. Han, "Collection of Schedule Quality Metrics and Application to Projects of the Office of Facilities Planning and Construction (OFPC) APPROVED BY SUPERVISING COMMITTEE :," 2015.

[21] R. J. Dzeng and H. Y. Lee, "Critiquing contractors' scheduling by integrating rule-based and case-based reasoning," *Automation in Construction*, vol. 13, no. 5 SPEC. ISS., pp. 665–678, sep 2004.

[22] S. Emdanat and M. Azambuja, "Aligning Near and Long Term Planning for Lps Implementations: a Review of Existing and New Metrics," no. 5, pp. 103–112, 2016.

[23] F. R. Hamzeh, G. El Samad, and S. Emdanat, "Advanced Metrics for Construction Planning," *Journal of Construction Engineering and Management*, vol. 145, no. 11, 2019. [Online]. Available: https://orcid.org/0000-0001-8158-2599.

[24] F. Hamzeh, G. Ballard, and I. D. Tommelein, "Rethinking lookahead planning to optimize construction workflow." *Lean Construction Journal*, 2012.

[25] J. J. Lin, K. k. Han, and M. Golparvar-Fard, "A Framework for Model-driven Acquisition and Analytics of Visual Data using UAVs for Automated Construction Progress Monitoring," in *2015 International Workshop on Computing in Civil Engineering*, 2015, pp. 1–4.

[26] J. J. Lin and M. Golparvar-Fard, "Visual data and predictive analytics for proactive project controls on construction sites," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10863 LNCS. Springer Verlag, 2018, pp. 412–430.

[27] L. Rizk, F. R. Hamzeh, and S. Emdanat, "Introducing new capacity planning metrics in production planning," *Proc. 25th Ann. Conf. of the Int'l. Group for Lean Construction*, no. July, pp. 679–686, 2017.

[28] O. Seppänen, R.-U. Modrich, and G. Ballard, "Integration of Last Planner System and Location-Based Management System," *23rd Annual Conference of the International Group for Lean Construction*, no. July, pp. 123–132, 2015. [Online]. Available: http://www.iglc.net/papers/details/1201

[29] Z. Abuwarda and T. Hegazy, "Multi-dimensional optimization model for schedule fast-tracking without over-stressing construction workers," *Canadian Journal of Civil Engineering*, no. ja, 2019.

[30] A. Altuwaim and K. El-Rayes, "Minimizing duration and crew work interruptions of repetitive construction projects," *Automation in Construction*, vol. 88, pp. 59–72, apr 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S092658051630379X

[31] A. C. Devi and K. Ananthanarayanan, "A model for automated generation of resource-driven construction schedule for housing industry," in *Automation and Robotics in Construction - Proceedings of the 24th International Symposium on Automation and Robotics in Construction*, 2007, pp. 421–427.

[32] K. El-Rayes and A. Kandil, "Time-Cost-Quality Trade-Off Analysis for Highway Construction," *Journal of Construction Engineering and Management*, vol. 131, no. 4, pp. 477–486, 2005. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%282005%29131%3A4%28477%29

[33] A. Kandil and K. El-Rayes, "Parallel Genetic Algorithms for Optimizing Resource Utilization in Large-Scale Construction Projects," *Journal of Construction Engineering and Management*, vol. 132, no. 5, pp. 491–498, 2006. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%282006%29132%3A5%28491%29

[34] F. Amer, H. Y. Koh, and M. Golparvar-Fard, "Automated Methods and Systems for Construction Planning and Scheduling: Critical Review of Three Decades of Research," *Journal of Construction Engineering and Management*, vol. 147, no. 7, pp. (ASCE)CO.1943–7862.0 002 093, jul 2021. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%29CO.1943-7862.0002093

[35] A. Desgagné-Lebeuf, N. Lehoux, R. Beauregard, and G. Desgagné-Lebeuf, "Computer-assisted scheduling tools in the construction industry: A systematic literature review," in *IFAC-PapersOnLine*, vol. 52, no. 13. Elsevier B.V., sep 2019, pp. 1843–1848.

[36] M. Fischer and F. Aalami, "Scheduling with Computer-Interpretable Construction Method Models," 1995. [Online]. Available: https://stacks.stanford.edu/file/druid:gw256py6336/WP041.pdf

[37] V. Faghihi, A. Nejat, K. F. Reinschmidt, and J. H. Kang, "Automation in construction scheduling: a review of the literature," *International Journal of Advanced Manufacturing Technology*, vol. 81, no. 9-12, pp. 1845–1856, 2015. [Online]. Available: https://link.springer.com/content/pdf/10.1007%2Fs00170-015-7339-0.pdf

[38] K. Xu and H. Muñoz-Avila, "CaBMA: A case-based reasoning system for capturing, refining, and reusing project plans," *Knowledge and Information Systems*, vol. 15, no. 2, pp. 215–232, 2008.

[39] H.-G. Ryu and H.-S. Lee, "Schedule Management Process Model Based on CSDM," in *Proceedings of the 21st ISARC*, Jeju, South Korea, 2004.

[40] McKinsey&Company, "The Construction Productivity imperative, retrieved from https://www.mckinsey.com/business-functions/operations/our-insights/the-construction-productivity-imperative on 06-11-2020," https://www.mckinsey.com/business-functions/operations/our-insights/the-construction-productivity-imperative, 2020.

[41] D. W. Halpin and B. A. Senior, *Construction management*. Wiley, 2011.

[42] S. M. E. Sepasgozar, R. Karimi, S. Shirowzhan, M. Mojtahedi, S. Ebrahimzadeh, and D. McCarthy, "Delay causes and emerging digital tools: A novel model of delay analysis, including integrated project delivery and pmbok," *Buildings*, vol. 9, no. 9, 2019. [Online]. Available: https://www.mdpi.com/2075-5309/9/9/191

[43] R. Charef, H. Alaka, and S. Emmitt, "Beyond the third dimension of bim: A systematic review of literature and assessment of professional views," *Journal of Building Engineering*, vol. 19, pp. 242–257, 2018. [Online]. Available: https://app.dimensions.ai/details/publication/pub.1103788372

[44] B. Naticchia, A. Carbonari, M. Vaccarini, and R. Giorgi, "Holonic execution system for real-time construction management," *Automation in Construction*, vol. 104, pp. 179–196, aug 2019.

[45] R. Volk, T. H. Luu, J. S. Mueller-Roemer, N. Sevilmis, and F. Schultmann, "Deconstruction project planning of existing buildings based on automated acquisition and reconstruction of building information," *Automation in Construction*, vol. 91, pp. 226–245, jul 2018.

[46] Y. Weldemihret Weldu, "Automated Generation and Visualization of Initial Construction Schedules from Building Information Models," Ph.D. dissertation, LSU, 2016. [Online]. Available: https://digitalcommons.lsu.edu/gradschool{_}dissertations/175

[47] C. E. Firat, D. Arditi, J. P. Hämäläinen, and J. Kiiras, "Extended model-based master scheduling for building projects using advanced line of balance," *Managing IT in Construction/Managing Construction for Tomorrow*, p. 85, 2009. [Online]. Available: http://itc.scix.net/data/works/att/w78-2009-1-29.pdf

[48] S. H. Lee, F. Peña-Mora, and M. Park, "Dynamic planning and control methodology for strategic and operational construction project management," *Automation in Construction*, vol. 15, no. 1, pp. 84–97, jan 2006.

[49] M. Mawdesley and W. Askew, "Automating project scheduling. A case study," in *Preparing for Construction in the 21st Century*. Publ by ASCE, 1991, pp. 360–365.

[50] A. Sirajuddin and M. J. Mawdesley, "A knowledge-based system for recognition of work required for construction," *Computers and Structures*, vol. 40, no. 1, pp. 37–44, 1991.

[51] M. Al-Shawi and P. Brandon, "An expert system to assist in generation and scheduling of construction activities," in *Building Economics and Construction Management, CIB 90 2, 373-384*, 1990.

[52] R. E. Levitt and J. Kunz, "USING ARTIFICIAL INTELLIGENCE TECHNIQUES TO SUPPORT PROJECT MANAGEMENT," Tech. Rep. 1, 1987.

[53] B. Bremdal, "LIFT: A knowledge based planning system," Div. of Marine Systems Design, Institute for Marin Prosjektering NTH, Oslo, Norway, Tech. Rep., 1986.

[54] R. Levitt and J. Kunz, "Using knowledge of construction and project management for automated schedule updating," *Project Management Journal*, vol. 16, no. 5, pp. 57–76, 1985.

[55] H. Liu, M. Al-Hussein, and M. Lu, "BIM-based integrated approach for detailed construction scheduling under resource constraints," *Automation in Construction*, vol. 53, pp. 29–43, 2015.

[56] S. M. Chen, F. H. Griffis, P. H. Chen, and L. M. Chang, "A framework for an automated and integrated project scheduling and management system," *Automation in Construction*, vol. 35, pp. 89–110, 2013.

[57] A. L. Stumpf, R. Ganeshan, S. Chin, and L. Y. Liu, "Object-oriented model for integrating construction product and process information," *Journal of Computing in Civil Engineering*, vol. 10, no. 3, pp. 204–212, 1996.

[58] C. Zozaya-Gorostiza, C. Hendrickson, and D. R. Rehak, "A knowledge-intensive planner for construction projects," *Building and Environment*, vol. 25, no. 3, pp. 269–278, 1990.

[59] C. Hendrickson, "An expert system architecture for construction planning," Carnegie Mellon University, Tech. Rep., 1986. [Online]. Available: http://repository.cmu.edu/cee/34papers3://publication/uuid/B05DF627-E710-44A8-83C1-B893F61DD344

[60] F. Aalami and M. Fischer, "Joint product and process model elaboration based on construction method models," *Cib Report*, pp. 1–12, 1998.

[61] J. K. W. Yeoh, T. Q. Nguyen, and E. L. S. Abbott, "Construction Method Models Using Context Aware Construction Requirements for Automated Schedule Generation," in *Computing in Civil Engineering*, 2017, pp. 60–67.

[62] D. K. Chua, T. Q. Nguyen, and K. W. Yeoh, "Automated construction sequencing and scheduling from functional requirements," *Automation in Construction*, vol. 35, pp. 79–88, 2013.

[63] W. Huhnt and F. Enge, "Can algorithms support the specification of construction schedules?" in *Electronic Journal of Information Technology in Construction*, vol. 11, 2006. [Online]. Available: http://www.itcon.org/2006/39/ pp. 547–564.

[64] N. Kartam, R. E. Levitt, and D. E. Wilkins, "EXTENDING ARTIFICIAL INTELLIGENCE TECHNIQUES FOR HIERARCHICAL PLANNING," *Journal of Computing in Civil Engineering*, 1991. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290887-3801%281991%295%3A4%28464%29

[65] N. A. Kartam, "Investigating the utility of artificial intelligence techniques for automatic generation of construction project plans," Ph.D. dissertation, Stanford, 1989.

[66] K. Kim and Y. K. Cho, "Construction-specific spatial information reasoning in Building Information Models," *Advanced Engineering Informatics*, vol. 29, no. 4, pp. 1013–1027, oct 2015.

[67] V. Faghihi, K. F. Reinschmidt, and J. H. Kang, "Construction scheduling using Genetic Algorithm based on Building Information Model," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7565–7578, nov 2014.

[68] T. Adjei-kumit and A. Retik, "A library-based 4D visualisation of construction processes - IEEE Conference Publication," in *IEEE Conference on Information Visualization*, 1997, pp. 315–321.

[69] R. Akbaş, "Geometry-Based Modeling and Simulation of Construction Processes," *CIFE Technical Report*, vol. #151, 2004. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.84.8062{&}rep=rep1{&}type=pdf

[70] Z. Wang and E. Rezazadeh Azar, "BIM-based draft schedule generation in reinforced concrete-framed buildings," *Construction Innovation*, vol. 19, no. 2, pp. 280–294, apr 2019.

[71] S. Mohammadi, M. Tavakolan, and B. Zahraie, "Automated Planning of Building Construction Considering the Amount of Available Floor Formwork," in *Construction Research Congress 2016: Old and New Construction Technologies Converge in Historic San Juan - Proceedings of the 2016 Construction Research Congress, CRC 2016*, 2016, pp. 2197–2206.

[72] N. Dong, M. Fischer, Z. Haddad, and R. Levitt, "A method to automate look-ahead schedule (LAS) generation for the finishing phase of construction projects," *Automation in Construction*, vol. 35, pp. 157–173, 2013. [Online]. Available: http://ac.els-cdn.com/S0926580513000903/1-s2.0-S0926580513000903-main.pdf?{_}tid=3fb6740a-4657-11e7-afc2-00000aab0f6c{&}acdnat=1496272800{_}6dbbcbfe91688ae00001012e6b2aa54f

[73] R. Volk, N. Sevilmis, and F. Schultmann, "Deconstruction project planning of existing buildings based on automated acquisition and reconstruction of building information," in *Smart and Sustainable Built Environment (SASBE) Conference*, University of Pretoria, Pretoria, South Africa, 2015.

[74] E. Mikulakova, M. König, E. Tauscher, and K. Beucke, "Knowledge-based schedule generation and evaluation," *Advanced Engineering Informatics*, vol. 24, pp. 389–403, 2010. [Online]. Available: https://ac.els-cdn.com/S1474034610000546/1-s2.0-S1474034610000546-main.pdf?{_}tid=4dc258fc-ecfb-4467-b2d3-e92b1c146703{&}acdnat=1550688289{_}e1fba55f9947afbd18ba9c1c347b0a1a

[75] E. Tauscher, E. Mikulakova, K. Beucke, and M. König, "Automated Generation of Construction Schedules Based on the IFC Object Model," in *Computing in Civil Engineering (2009)*, 2009. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/41052%28346%2966http://ascelibrary.org/doi/10.1061/41052%28346%2966 pp. 666–675.

[76] C. E. Firat, J. Kiiras, and P. Huovinen, "Solving fundamental problems in model based, semi-automated building project scheduling," in *International Multi-Conference on Engineering and Technological Innovation, Proceedings*, vol. 1, 2008. [Online]. Available: http://www.iiis.org/CDs2008/CD2008SCI/CITSA2008/PapersPdf/I021SK.pdfhttp://www.scopus.com/inward/record.url?eid=2-s2.0-84893173581{&}partnerID=tZOtx3y1 pp. 76–80.

[77] S. Q. Wang, "ESSCAD : EXPERT SYSTEM INTEGRATING CONSTRUCTION SCHEDULING WITH CAD DRAWING," in *Proceedings of the CIB W78 International Conference on IT in Construction*, 2001. [Online]. Available: https://pdfs.semanticscholar.org/08a9/dff74507c0b04423324f1387e370d848c503.pdf pp. 1–14.

[78] O. Moselhi and M. J. Nicholas, "Hybrid Expert System for Construction Planning and Scheduling," *Journal of Construction Engineering and Management*, vol. 116, no. 2, pp. 221–238, 1990.

[79] B. De Vries and J. M. Harink, "Generation of a construction planning from a 3D CAD model," *Automation in Construction*, vol. 16, no. 1, pp. 13–18, 2007. [Online]. Available: www.elsevier.com/locate/autcon

[80] V. Hartmann, K. E. Beucke, K. Shapir, and M. König, "Model-based Scheduling for Construction Planning," *14th International Conference on Computing in Civil and Building Engineering*, 2012.

[81] G. Winstanley, M. A. Chacon, and R. E. Levitt, "Model-based planning: Scaled-up construction application," *Journal of Computing in Civil Engineering*, vol. 7, no. 2, pp. 199–217, 1993.

[82] J. Cherneff, R. Logcher, and D. Sriram, "Integrating CAD with Construction-Schedule Generation," *Journal of Computing in Civil Engineering*, vol. 5, no. 1, pp. 64–84, 1991. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290887-3801%281991%295%3A1%2864%29

[83] D. K. Chua, T. Q. Nguyen, and K. W. Yeoh, "Automated construction sequencing and scheduling from functional requirements," *Automation in Construction*, vol. 35, pp. 79–88, 2013. [Online]. Available: http://dx.doi.org/10.1016/j.autcon.2013.03.002

[84] O. Haiati, J. Von Heyl, and S. Schmalz, "BIM and sequence simulation in structural work - Development of a procedure for automation," in *IGLC 2016 - 24th Annual Conference of the International Group for Lean Construction*, 2016. [Online]. Available: www.iglc.net pp. 73–82.

[85] J. Park and H. Cai, "Automatic construction schedule generation method through BIM model creation," in *Congress on Computing in Civil Engineering, Proceedings*, vol. 2015-Janua, no. January, 2015, pp. 620–627.

[86] H. Kim, K. Anderson, S. Lee, and J. Hildreth, "Generating construction schedules through automatic data extraction using open BIM (building information modeling) technology," *Automation in Construction*, vol. 35, pp. 285–295, 2013.

[87] I. C. Wu, A. Borrmann, U. Beißert, M. König, and E. Rank, "Bridge construction schedule generation with pattern-based construction methods and constraint-based simulation," in *Advanced Engineering Informatics*, vol. 24, no. 4, nov 2010, pp. 379–388.

[88] H.-G. Ryu, H.-S. Lee, and M. Park, "Construction Planning Method Using Case-Based Reasoning (CONPLA-CBR)," *Journal of Computing in Civil Engineering*, vol. 21, no. 6, pp. 410–422, nov 2007. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%290887-3801%282007%2921%3A6%28410%29

[89] M. Alshawi and D. Jagger, "AN EXPERT SYSTEM TO ASSIST IN GENERATING AND SCHEDULING CONSTRUCTION ACTIVITIES," Tech. Rep. I.

[90] O. Shaked and A. Warszawski, "Consched: Expert system for scheduling of modular construction projects," *Journal of Construction Engineering and Management*, vol. 118, no. 3, pp. 488–506, 1992.

[91] R. Sacks and A. Warszawski, "A project model for an automated building system: Design and planning phases," *Automation in Construction*, vol. 7, no. 1, pp. 21–34, dec 1997.

[92] O. Shaked and A. Warszawski, "Knowledge-Based System for Construction Planning of High-Rise Buildings." *Journal of Construction Engineering & Management*, vol. 121, no. 2, p. 172, 1995. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%281995%29121%3A2%28172%29

[93] N. A. Kartam, "ISICAD: Interactive System for Integrating CAD and Computer-Based Construction Systems," *Computer-Aided Civil and Infrastructure Engineering*, vol. 9, no. 1, pp. 41–51, jan 1994. [Online]. Available: http://doi.wiley.com/10.1111/j.1467-8667.1994.tb00360.x

[94] D. Echeverry, "Factors for Generating Initial Construction Schedules," 1991.

[95] N. P. Garcia-Lopez and M. Fischer, "A Construction Workflow Model for Analyzing the Impact of In-Project Variability," *Construction Research Congress 2016: Old and New Construction Technologies Converge in Historic San Juan - Proceedings of the 2016 Construction Research Congress, CRC 2016*, no. November, pp. 1998–2007, 2016.

[96] E. Tauscher, E. Mikulakova, M. König, and K. Beucke, "Generating Construction Schedules with Case-Based Reasoning Support," Tech. Rep., 2007. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/40937%28261%2915

[97] E. Tauscher, K. Smarsly, M. König, and K. Beucke, "Automated generation of construction sequences using building information models," in *Computing in Civil and Building Engineering - Proceedings of the 2014 International Conference on Computing in Civil and Building Engineering*. American Society of Civil Engineers (ASCE), 2014, pp. 745–752.

[98] K. J. Lee, H. W. Kim, J. K. Lee, and T. H. Kim, "Case- and Constraint-Based Project Planning for Apartment Construction," *AI Magazine*, vol. 19, no. 1, pp. 13–13, mar 1998.

[99] M. König, C. Koch, I. Habenicht, and S. Spieckermann, "Intelligent BIM-based construction scheduling using discrete event simulation," *Proceedings - Winter Simulation Conference*, no. December, 2012.

[100] M. Kataoka, "Automated Generation of Construction Plans from Primitive Geometries," *JOURNAL OF CONSTRUCTION ENGINEERING AND MANAGEMENT*, vol. 134, no. 8, pp. 592–600, 2008. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%282008%29134%3A8%28592%29

[101] I. D. Tommelein, R. I. Carr, and A. M. Odeh, "Knowledge-based Assembly of Simulation Networks Using Construction Designs, Plans, and Methods," in *Proceedings of the 26th Conference on Winter Simulation*, 1994. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%281994%29120%3A4%28796%29http://dl.acm.org/citation.cfm?id=193201.194684 pp. 1145–1152.

[102] A. Morad and Y. Beliveau, "Geometric-based reasoning system for project planning," *Journal of Computing in Civil Engineering*, vol. 9, no. 4, pp. 293–294, 1994.

[103] C. K. Soh and T. J. Lee, "Approach to automate the generation of construction schedules," in *Artificial Intelligence and Civil Engineering*. Publ by Civil-Comp Limited, 1991, pp. 57–64.

[104] N. Kano, "A knowledge-based system for construction planning and scheduling," in *Proc. 7th Int. Symp. on Automation and Robotics in Constr*, 1990.

[105] C. Zozaya-Gorostiza, C. Hendrickson, D. R. Rehak, and P. Lim, "Construction PLANEX: A Knowledge Intensive Planner for Construction Projects," in *The 5th International Symposium on Robotics in Construction June 6-8, Tokyo*, 1988.

[106] K. McKinney and M. Fischer, "Generating, evaluating and visualizing construction schedules with CAD tools," *Automation in Construction*, vol. 7, no. 6, pp. 433–447, 1998.

[107] J. H. Tah, V. Carr, and R. Howes, "Information modelling for case-based construction planning of highway bridge projects," *Advances in engineering software*, vol. 30, no. 7, pp. 495–509, jul 1999.

[108] R.-J. Dzeng and I. D. Tommelein, "Boiler Erection Scheduling Using Product Models and Case-Based Reasoning," *Journal of Construction Engineering and Management*, vol. 123, no. 3, pp. 338–347, 1997. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%281997%29123%3A3%28338%29

[109] N.-J. Yau, "An object-oriented project model for integrating building design, construction scheduling, and cost estimating for mid-rise construction," Tech. Rep., 1992.

[110] H. Rankin, J., T. M. Froese, and L. M. Waugh, "EXPLORING THE APPLICATION OF CASE-BASED REASONING TO COMPUTER-ASSISTED CONSTRUCTION PLANNING," in *Durability of Building Materials and Components 8.*, M.A. Lacasse and D. Vanier., Eds. Ottawa: Institute for Research in Construction, 1999. [Online]. Available: http://itc.scix.net/ pp. 2526–2536.

[111] C. O. Benjamin, D. L. Babcock, N. B. Yunus, and J. Kincaid, "Knowledge-based prototype for improving scheduling productivity," *Journal of computing in civil engineering*, vol. 4, no. 2, pp. 124–134, 1990. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290887-3801%281990%294%3A2%28124%29

[112] K. Kahkonen and P. Laurikka, "A Computer Program for Generating and Managing Schedules for Construction Projects," in *Proceedings of the 7th International Symposium on Automation and Robotics in Construction*, Bristol, England, 1990.

[113] D. Navinchandra, D. Sriram, and R. D. Logcher, "GHOST: Project network generator," *Journal of Computing in Civil Engineering*, vol. 2, no. 3, pp. 239–254, 1988.

[114] A. Morad and Y. Beliveau, "Knowledge-based planning system," *Journal of Construction Engineering and Management*, vol. 117, no. 1, pp. 1–12, 1991. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9364%281991%29117%3A1%281%29

[115] L. M. Waugh, "A Construction Planner," Ph.D. dissertation, Stanford University, 1990. [Online]. Available: https://cife.stanford.edu/sites/default/files/TR032.pdf

[116] G. Winstanley and K. Hoshi, "Model-Based Planning Utilizing Activity Aggregation Based on Zones," 1992. [Online]. Available: https://stacks.stanford.edu/file/druid:bq464mv1175/TR069.pdf

[117] C. W. Feng and Y. J. Chen, "Using MD CAD objects to automatically generate a contract-driven schedule," in *2006 Proceedings of the 23rd International Symposium on Robotics and Automation in Construction, ISARC 2006*, 2006, pp. 740–745.

[118] T. H. Nguyen, "Automated construction planning for multi-story buildings," in *Construction Research Congress 2005: Broadening Perspectives - Proceedings of the Congress*, 2005, pp. 1213–1222.

[119] R. Y. Huang and K. S. Sun, "System development for non-unit based repetitive project scheduling," *Automation in Construction*, vol. 14, no. 5, pp. 650–665, oct 2005.

[120] F. Aalami, J. Kunz, and M. Fischer, "Model-Based Mechanisms for Automated Activity Sequencing," 1998. [Online]. Available: http://cife.stanford.edu/online.publications/WP050.pdf

[121] J. H. Rankin, T. M. Froese, and L. M. Waugh, "Computer assisted construction planning (CACP) in the context of total project systems (TOPS)," in *Proceedings, Annual Conference - Canadian Society for Civil Engineering*, vol. 2, 1997, pp. 41–50.

[122] M. A. Fischer and F. Aalami, "Scheduling with Computer-Interpretable Construction Method Models," *Journal of Construction Engineering and Management*, vol. 122, no. 4, pp. 337–347, 1996.

[123] K. E. Kähkönen, "Interactive decision support system for building construction scheduling," *Journal of Computing in Civil Engineering*, vol. 8, no. 4, pp. 519–535, 1994.

[124] M. R. Mcgartland, C. T. Hendrickson, and A. M. Asce, "EXPERT SYSTEMS FOR CONSTRUCTION PROJECT MONITORING," *J. Constr. Eng. Manage*, vol. 111, no. 3, pp. 293–307, 1985.

[125] L. Wang and F. Leite, "Comparison of Experienced and Novice BIM Coordinators in Performing Mechanical, Electrical and Plumbing (MEP) Coordination Tasks," in *Construction Research Congress 2014*, 2014. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/9780784413517.003 pp. 21–30.

[126] W. C. Wang, S. W. Weng, S. H. Wang, and C. Y. Chen, "Integrating building information models with construction process simulations for project scheduling support," *Automation in Construction*, vol. 37, pp. 68–80, 2014.

[127] M. Fischer, F. Alami, and M. O'Brian Evans, "Model-Based Constructibility Analysis: The MOCA System," in *CIB W78 workshop on computer integrated construction*, Helsinki, 1994. [Online]. Available: https://stacks.stanford.edu/file/druid:fb274qm9235/WP034.pdf

[128] M. S. El-Abbasy, A. Elazouni, and T. Zayed, "MOSCOPEA: Multi-objective construction scheduling optimization using elitist non-dominated sorting genetic algorithm," *Automation in Construction*, vol. 71, no. Part 2, pp. 153–170, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.autcon.2016.08.038

[129] K. Hyari and K. El-Rayes, "Optimal planning and scheduling for repetitive construction projects," *Journal of Management in Engineering*, vol. 22, no. 1, pp. 11–19, 2006.

[130] J. Zhou, P. E. D. Love, X. Wang, K. L. Teo, and Z. Irani, "A review of methods and algorithms for optimizing construction scheduling," *Journal of the Operational Research Society*, vol. 64, no. 8, pp. 1091–1105, 2013. [Online]. Available: http://link.springer.com/10.1057/jors.2012.174

[131] A. Kandil, K. El-Rayes, and O. El-Anwar, "Optimization Research: Enhancing the Robustness of Large-Scale Multiobjective Optimization in Construction," *Journal of Construction Engineering and Management*, vol. 136, no. 1, pp. 17–25, 2010. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%29CO.1943-7862.0000140

[132] B. Naticchia, A. Carbonari, and M. Vaccarini, "A simulation-based approach for optimal construction planning and scheduling," in *Proceedings of the Creative Construction Conference*, Ljubljana, Slovenia, 2018, pp. 739–746.

[133] R. Volk, T. H. Luu, J. S. Mueller-Roemer, N. Sevilmis, and F. Schultmann, "Deconstruction project planning of existing buildings based on automated acquisition and reconstruction of building information," *Automation in Construction*, vol. 91, no. February, pp. 226–245, 2018. [Online]. Available: https://doi.org/10.1016/j.autcon.2018.03.017

[134] E. Tauscher, K. Smarsly, M. König, and K. Beucke, "Automated Generation of Construction Sequences using Building Information Models," in *2014 International Conference on Computing in Civil and Building Engineering*, 2014.

[135] J. K. W. Yeoh, T. Q. Nguyen, and E. L. S. Abbott, "Construction Method Models Using Context Aware Construction Requirements for Automated Schedule Generation," in *Computing in Civil Engineering*, 2017. [Online]. Available: http://ascelibrary.org/doi/pdf/10.1061/9780784480847.008

[136] F. Amer and M. Golparvar-Fard, "Automatic Understanding of Construction Schedules: Part-of-Activity Tagging," *Proceedings of the 2019 European Conference for Computing in Construction*, vol. 1, pp. 190–197, 2019.

[137] F. Amer and M. Golparvar-Fard, "Formalizing Construction Sequencing Knowledge and Mining Company-Specific Best Practices from Past Project Schedules," in *Computing in Civil Engineering 2019: Visualization, Information Modeling, and Simulation - Selected Papers from the ASCE International Conference on Computing in Civil Engineering 2019*, 2019, pp. 215–223.

[138] X. Zhao, K.-W. Yeoh, D. Kim, H. Chua, X. Zhao, K.-W. Yeoh, ·. D. K. H. Chua, and D. K. H. Chua, "Extracting Construction Knowledge from Project Schedules Using Natural Language Processing," in *Lecture Notes in Mechanical Engineering*, 2020. [Online]. Available: https://doi.org/10.1007/978-981-15-1910-9{_}17

[139] F. Amer and M. Golparvar-Fard, "Modeling dynamic construction work template from existing scheduling records via sequential machine learning," *Advanced Engineering Informatics*, vol. 47, jan 2021.

[140] M. Bügler and A. Borrmann, *Simulation Based Construction Project Schedule Optimization: An Overview on the State-of-the-Art*. IGI Global, (2016), 482–507.

[141] D.-E. Lee and D. Arditi, "Automated statistical analysis in stochastic project scheduling simulation," *Journal of Construction Engineering and Management*, vol. 132, no. 3, pp. 268–277, 2006.

[142] G. Ma, L. Gu, and N. Li, "Scenario-based proactive robust optimization for critical-chain project scheduling," *Journal of Construction Engineering and Management*, vol. 141, no. 10, 2015.

[143] K. Kim, J. Walewski, and Y. K. Cho, "Multiobjective construction schedule optimization using modified niched pareto genetic algorithm," *Journal of Management in Engineering*, vol. 32, no. 2, 2015.

[144] T. Salama and O. Moselhi, "Multi-objective optimization for repetitive scheduling under uncertainty," *Engineering, Construction and Architectural Management*, pp. 1294–1320, 2019.

[145] D. Heon Jun and K. El-Rayes, "Multiobjective optimization of resource leveling and allocation during construction scheduling," *Journal of construction engineering and management*, vol. 137, no. 12, pp. 1080–1088, 2011.

[146] N. Dong, D. Ge, M. Fischer, and Z. Haddad, "A genetic algorithm-based method for look-ahead scheduling in the finishing phase of construction projects," *Advanced Engineering Informatics*, vol. 26, no. 4, pp. 737–748, 2012.

[147] T. Hegazy, M. Said, and M. Kassab, "Incorporating rework into construction schedule analysis," *Automation in construction*, vol. 20, no. 8, pp. 1051–1059, 2011.

[148] V. Faghihi, A. Nejat, K. F. Reinschmidt, and J. H. Kang, "Automation in construction scheduling: a review of the literature," *The International Journal of Advanced Manufacturing Technology*, vol. 81, no. 9-12, pp. 1845–1856, 2015.

[149] S. Abourizk, D. Halpin, Y. Mohamed, and U. Hermann, "Research in modeling and simulation for improving construction engineering operations," in *Journal of Construction Engineering and Management*, vol. 137, no. 10, 2011. [Online]. Available: www.draxware.com pp. 843–852.

[150] N. A. Kartam, A. Member, and R. E. Levitt, "INTELLIGENT PLANNING OF CONSTRUCTION PROJECTS," *Journal of Computing in Civil Engineering*, vol. 4, no. 2, pp. 155–176, 1990. [Online]. Available: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290887-3801%281990%294%3A2%28155%29

[151] N. P. Garcia-Lopez, "An Activity And Flow-based Construction Model for Managing On-Site Work," Ph.D. dissertation, 2017. [Online]. Available: http://purl.stanford.edu/jw488vx5801

[152] L. Koskela, "APPLICATION OF THE NEW PRODUCTION PHILOSOPHY TO CONSTRUCTION," 1992.

[153] L. M. Waugh and T. M. Froese, "Constraint knowledge For Construction Scheduling," in *First International Conference on Expert Planning Systems*, 1991, pp. 114–118.

[154] K. Sigalov and M. König, "Recognition of process patterns for BIM-based construction schedules," *Advanced Engineering Informatics*, vol. 33, pp. 456–472, aug 2017.

[155] "Construction Planning Meets Artificial Intelligence, ALICE Technologies," https:www.alicetechnologies.com, Accessed May 2020.

[156] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations ofwords and phrases and their compositionality," in *Advances in Neural Information Processing Systems (2013)*, 2013. [Online]. Available: https://arxiv.org/pdf/1310.4546.pdf

[157] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Qatar, 2014. [Online]. Available: http://nlp. pp. (2014) 1532–1543.

[158] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017. [Online]. Available: http://www.isthe.com/chongo/tech/comp/fnv

[159] A. J. P. Tixier, M. Vazirgiannis, and M. R. Hallowell, "Word Embeddings for the Construction Domain, arXiv:1610.09333v1," 2016. [Online]. Available: https://radimrehurek.com/gensim/models/word2vec.http://arxiv.org/abs/1610.09333

[160] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2015, 3rd International Conference for Learning Representations. [Online]. Available: http://arxiv.org/abs/1412.6980

[161] T. Tieleman and G. Hinton, "Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude," Coursera: Neural Networks for Machine Learning, 2012.

[162] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," pp. 1–9, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[163] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," 2014.

[164] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[165] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html

[166] "Embedding projector - visualization of high-dimensional data, https://projector.tensorflow.org/," 2019.

[167] F. Amer, J. Hockenmaier, and M. Golparvar-Fard, "Learning and critiquing pairwise activity relationships for schedule quality control via deep learning-based natural language processing," *Automation in Construction*, p. 104036, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580521004878

[168] S. Changali, A. Mohammad, and M. Van Nieuwland, "The construction productivity imperative," Tech. Rep. June, 2015.

[169] PMI, *Practice Standard for Scheduling - Third Edition*, 2019. [Online]. Available: https://www.pmi.org/pmbok-guide-standards/framework/practice-standard-scheduling-3rdedition

[170] Defense Contract Management Agency, "Earned Value Management System (EVMS) Program Analysis Pamphlet (PAP)," Tech. Rep., 2012.

[171] Government Accountability Office, *Schedule Assessment Guide: Best Practices for project schedules*, 2015, no. May 2012. [Online]. Available: http://www.gao.gov/products/GAO-16-89G

[172] National Defense Industrial Association, "Planning & Scheduling Excellence Guide (PASEG)," Tech. Rep., 2019. [Online]. Available: www.ndia.org

[173] H. Emam, S. M. Ahmed, and P. Farrell, "Linear Schedules Quality Assssment : A Framework for Construction Projects with Repetitive Activities A Framework for Construction Projects with Repetitive Activities," no. December, 2014.

[174] A. Kavad, R. Dharsandia, A. Hosny, and M. Nik-Bakht, "Schedule quality assessment for nd models using industry foundation classes," in *Proceedings of the 36th International Symposium on Automation and Robotics in Construction, ISARC 2019*, 2019, pp. 1050–1056.

[175] J. M. De La Garza and C. W. Ibbs, "Knowledge-Elicitation Study in Construction Scheduling Domain," *Journal of Computing in Civil Engineering*, vol. 4, no. 2, pp. 135–153, apr 1990. [Online]. Available: http://ascelibrary.org/doi/10.1061/%28ASCE%290887-3801%281990%294%3A2%28135%29

[176] R. J. Dzeng and H. Y. Lee, "Critiquing contractors' scheduling by integrating rule-based and case-based reasoning," *Automation in Construction*, vol. 13, no. 5 SPEC. ISS., pp. 665–678, 2004.

[177] A. D. Russell and A. Udaipurwala, "Assessing the quality of a construction schedule," in *Proceedings of Construction Congress VI: Building Together for a Better Tomorrow in an Increasingly Complex World*, vol. 278, 2000, pp. 928–937.

[178] V. K. Bansal and M. Pal, "Generating, Evaluating, and Visualizing Construction Schedule with Geographic Information Systems," *Journal of Computing in Civil Engineering*, vol. 22, no. 4, pp. 233–242, 2008.

[179] V. K. Bansal and M. Pal, "Construction schedule review in GIS with a navigable 3D animation of project activities," *International Journal of Project Management*, vol. 27, no. 5, pp. 532–542, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.ijproman.2008.07.004

[180] R. K. Soman, M. Molina-Solana, and J. K. Whyte, "Linked-Data based Constraint-Checking (LDCC) to support look-ahead planning in construction," *Automation in Construction*, vol. 120, dec 2020. [Online]. Available: https://doi.org/10.1016/j.autcon.2020.103369

[181] M. Bilal, L. O. Oyedele, J. Qadir, K. Munir, S. O. Ajayi, O. O. Akinade, H. A. Owolabi, H. A. Alaka, and M. Pasha, "Big data in the construction industry: A review of present status, opportunities, and future trends," *Advanced engineering informatics*, vol. 30, no. 3, pp. 500–521, 2016.

[182] J. Zhang and 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.

[183] T. Kim and S. Chi, "Accident case retrieval and analyses: Using natural language processing in the construction industry," *Journal of Construction Engineering and Management*, vol. 145, no. 3, pp. pp. 04 019 004, doi=10.1061/(ASCE)CO.1943–7862.0 001 625, 2019. [Online]. Available: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%29CO.1943-7862.0001625

[184] P. Zhou and 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 p 1669*, vol. 1676, 2016.

[185] D. Guo, E. Onstein, and A. D. La Rosa, "A semantic approach for automated rule compliance checking in construction industry," *IEEE Access*, vol. 9, pp. pp. 129 648 – 129 660, doi:10.1109/ACCESS.2021.3 108 226, 2021.

[186] J.-R. Lin, Z.-Z. Hu, J.-P. Zhang, and F.-Q. Yu, "A natural-language-based approach to intelligent data retrieval and representation for cloud bim," *Computer-Aided Civil and Infrastructure Engineering*, vol. 31, no. 1, pp. 18–33, 2016.

[187] H. Alikhani, C. Le, and 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, July 2020, pp. 2–6.

[188] X. Zhao, K.-W. Yeoh, and D. K. H. Chua, "Extracting construction knowledge from project schedules using natural language processing," in *The 10th International Conference on Engineering, Project, and Production Management*. Springer, 2020, pp. 197–211.

[189] G. Ballard, "THE LAST PLANNER SYSTEM OF PRODUCTION CONTROL," Ph.D. dissertation, The University of Birmingham, 2000.

[190] M. W. Chang, L. Ratinov, and D. Roth, "Structured learning with constrained conditional models," *Machine Learning*, vol. 88, no. 3, pp. 399–431, sep 2012.

[191] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[192] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2015. [Online]. Available: https://arxiv.org/abs/1412.6980v9

[193] F. Amer, Y. Jung, and M. Golparvar-Fard, "Transformer machine learning language model for auto-alignment of long-term and short-term plans in construction," *Automation in Construction*, vol. 132, p. 103929, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0926580521003800

[194] H. G. Ballard, "The last planner system of production control," Ph.D. dissertation, University of Birmingham, 2000.

[195] R. Sacks, O. Seppänen, V. Priven, and J. Savosnick, "Construction flow index: a metric of production flow quality in construction," *Construction Management and Economics*, vol. 6193, no. January, pp. 1–19, 2017. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/01446193.2016.1274417

[196] F. R. Hamzeh, G. Ballard, and 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.

[197] G. Ballard and G. Howell, "An update on last planner," in *Proc., 11th Annual Conf., International Group for Lean Construction, Blacksburg, VA*, 2003.

[198] J. L. Fernandez-Solis, V. Porwal, S. Lavy, A. Shafaat, Z. K. Rybkowski, K. Son, and N. Lagoo, "Survey of motivations, benefits, and implementation challenges of last planner system users," *Journal of construction engineering and management*, vol. 139, no. 4, pp. 354–360, 2013.

[199] B. Dave, J.-P. Hämäläinen, L. Koskela et al., "Exploring the recurrent problems in the last planner implementation on construction projects," in *Proceedings of the Indian Lean Construction Conference (ILCC 2015)*. Institute for Lean Construction Excellence, 2015.

[200] J. J. Lin and M. Golparvar-Fard, "Construction progress monitoring using cyber-physical systems," in *Cyber-Physical Systems in the Built Environment*. Springer, 2020, pp. 63–87.

[201] J. J. Lin and M. Golparvar-Fard, "A visual and virtual production management system for proactive project controls," 2021, (accepted).

[202] X. Wu and M. Lode, "Language Models are Unsupervised Multitask Learners (Summarization)," *OpenAI Blog*, vol. 1, no. May, pp. 1–7, 2020. [Online]. Available: https://github.com/codelucas/newspaper

[203] B. Dave and R. Sacks, "Production control systems for construction at the nexus of lean and bim," *Lean Construction: Core Concepts and New Frontiers, New York: Routledge*, pp. 54–83, 2020.

[204] G. Ballard and I. Tommelein, "Current process benchmark for the last planner system," *Lean Construction Journal*, vol. 89, pp. 57–89, 2016.

[205] R. Sacks, "What constitutes good production flow in construction?" *Construction Management and Economics*, vol. 34, no. 9, pp. 641–656, 2016. [Online]. Available: http://www.tandfonline.com/doi/full/10.1080/01446193.2016.1200733

[206] D. Heigermoser, B. G. de Soto, E. L. S. Abbott, and D. K. H. Chua, "Bim-based last planner system tool for improving construction project management," *Automation in Construction*, vol. 104, pp. 246–254, 2019.

[207] M. Kopsida, I. Brilakis, and 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, 27-29 October*, 2015, pp. 421–431.

[208] B. Ekanayake, J. K.-W. Wong, A. A. F. Fini, and P. Smith, "Computer vision-based interior construction progress monitoring: A literature review and future research directions," *Automation in Construction*, vol. 127, p. 103705, 2021.

[209] T. Sawyer, "Modeling supply chains," vol. 260, no. 14, pp. pp. 24–27, 04 2008. [Online]. Available: http://worldcat.org/issn/08919526(visitedon2021-08-21)

[210] P. Jafari, M. Al Hattab, E. Mohamed, and S. AbouRizk, "Automated extraction and time-cost prediction of contractual reporting requirements in construction using natural language processing and simulation," *Applied Sciences*, vol. 11, no. 13, 2021.

[211] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, Ł. Kaiser, and N. Shazeer, "Generating wikipedia by summarizing long sequences," *arXiv*, pp. 1–18, 2018.

[212] M. E. Peters, M. Neumann, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," Tech. Rep., 2018. [Online]. Available: http://allennlp.org/elmo

[213] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the Limits of Language Modeling," 2016. [Online]. Available: http://arxiv.org/abs/1602.02410

[214] S. Merity, N. Shirish Keskar, and R. Socher, "Regularizing and Optimizing LSTM Language Models," Tech. Rep., 2017.

[215] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," Tech. Rep., 2013. [Online]. Available: http://ronan.collobert.com/senna/

[216] D. Bahdanau, K. H. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[217] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving Language Understanding by Generative Pre-Training," 2018. [Online]. Available: https://gluebenchmark.com/leaderboard

[218] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," 2018. [Online]. Available: https://github.com/tensorflow/tensor2tensorhttp://arxiv.org/abs/1810.04805

[219] A. Fan, M. Lewis, and Y. Dauphin, "Hierarchical neural story generation," Tech. Rep., 2018. [Online]. Available: www.github.com/pytorch/fairseq

[220] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2019. [Online]. Available: https://github.com/ari-holtzman/degen

[221] B. E. Levitt, N. A. Kartam, and J. C. Kunz, "ARTIFICIAL INTELLIGENCE TECHNIQUES FOR GENERATING CONSTRUCTION PROJECT PLANS," *Journal of Construction Engineering and Management*, vol. 114, no. 3, 1988.