AUTOMATED CONSTRUCTION WORK PACKAGE DECISION-MAKING MODEL USING RSMEANS AND LSTM BASED DEEP LEARNING

A Thesis

by

JOHN SEOK OH

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2022

Major Subject: Construction Management

ABSTRACT

Construction projects involve complicated workflows with efficient resource management. One of the methods for successful project planning is Work Packaging (WP.) Composition of the WP has been done by human understanding, resulting in several critical problems, such as challenging WP of the inexperienced, format and term inconsistency, and time-consuming and error-prone nature of the human activity. Thus, the WP assistant system is in need to solve the problems. This study aims to develop an automated WP decision-making prototype from detail section drawings. The research objectives are as follows: 1) Organize detail section drawings using RSMeans Assemblies Costs standard, 2) Construct prototype which automates WP decision procedure, 3) Evaluate the sensitivity of the decision output. This prototype produced about 95.24% testing accuracy from 314 datasets, significantly accurate for automated WP decisions. This research is expected to solve problems from human WP compositions and eventually contribute to the efficient WP organization of construction project entities.

DEDICATION

To my beloving family, Dong Ik Oh and Heajong Shin,

who provides endless support, inspiration, and love.

ACKNOWLEDGEMENTS

First, I would like to show sincere gratitude to my committee chair, Dr. Ryoo, committee co-chair, Dr. Behzadan, and my committee member, Dr. Huang, who have been supportive throughout my entire coursework. They tried their best to help conduct research and provide precious advice without hesitation.

I would also like to thank the A&M Korean Student Church members, AKOZ, and Department of Construction Science graduate students. They have prayed for my success and provided the precious experience at Texas A&M.

Thanks also go to my family members and friends outside Texas. They have sent me their encouragement, and I can withstand difficult times.

Finally, I appreciate my parents, who have supported, taught, and loved me for my entire life.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professor Boong Yeol Ryoo[advisor] and Professor Amir Behzadan[co-advisor] of the Department of Construction Science and Professor Ruihong Huang of the Department of Computer Science and Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

A scholarship from Texas A&M University Department of Construction Science Graduate School supported the graduate study.

This work was also made possible in part of Dynamic Systems Inc. Endowed Fellowship. Its contents are solely responsible for the author and do not necessarily represent the official views of the Dynamic System Inc.

NOMENCLATURE

IPA Independent Project Analysis

PM Project Management

WP Work Packaging

WBS Work Breakdown Structure

CWP Construction Work Package

AEC Architecture, Engineering, and Construction

CEM Construction Engineering, and Management

ML Machine Learning

AI Artificial Intelligence

DL Deep Learning

PMBOK Project Management Body of Knowledge

PMI Project Management Institute

ASTM American Society for Testing and Material

NLP Natural Language Processing

GAN Generative Adversarial Network

CNN Convolutional Neural Network

LSTM Long-Short Term Memory

RNN Recurrent Neural Network

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

Page

CHAPTER I

INTRODUCTION

Problem

Construction projects consist of complex works which involve various kinds of resources. In order to complete the project successfully within limited resources, plans for proper allocation of work and resources are necessary. According to the Independent Project Analysis (IPA) study, over 33% of site-based projects fail to accomplish project objectives (Merrow, 2011). It was found out that such failures are highly related to the weak base of resource planning (Kim & Gibson, 2002) because the deficiency leads to unreliable resource estimates on complicated projects (Bosch-Rekveldt et al., 2011). One of the methods for efficient work planning is using Work Packaging (WP). WP, a well-known methodology or in the Project Management (PM) theory, is done by breaking down the scope of work into executable work units (Calabrese et al., 2019). When designing Work Breakdown Structure (WBS), construction drawings such as detail section drawings are examined to define the scope of work, followed by WP decision.

Such procedures have been done by human activity, which displays several limitations. First, WP requires experience and prior knowledge of the construction work process and WBS. In other words, people who lack such information, or even some project managers, do not have a clear understanding of WBS composition methodology (Jones, 2007). Moreover, as this process relies on the understandings of different individuals, the WP work format and convention are inconsistent among workers. According to The State of Project Management 2021 Annual Report by Wellington, about 25% of 214 project practitioner respondents pointed out that inconsistent work in PM is the most significant challenge in the industry, which is the 4th largest of all categories (Hines, 2021). Besides, the complexity of the project gradually increases(Williams, 2005), leading to a proportional boost of construction drawings amount. Processing such documents by humans is time-consuming and error-prone. Thus, a Construction Work Package (CWP) assistant system for inexperienced people, which produces coherent output with less time and error, is required.

Research Goal, Question, Objectives, and Plan

The goal of this research is to develop a prototype which automatically decides WP from detail section drawings according to standardized criteria.

The goal of this research encounters three significant challenges.

- How should detail section drawing data be organized?
- How should the automatic WP decision prototype be constructed?
- Will the prototype produce a significantly accurate decision?

The objectives to solve each research questions are as follows. Specifically, research plans are given to accomplish each objective.

- Objective 1: Organize detail section drawing data according to the RSMeans Assemblies Cost criteria

    - Plan 1: Collect detail section drawings and text data from the commercial construction projects
    - Plan 2: Analyze the characteristic of drawing and text data
- Objective 2: Build up the structure of the automated WP decision prototype
    - Plan 3: Construct the text extraction model
    - Plan 4: Construct the text classification model
- Objective 3: Evaluate the decision result to validate if the prototype produces the significantly accurate decision
    - Plan 5: Train and validate the model
    - Plan 6: Analyze the WP decision prototype output

The hierarchy of research goal, objective, and plan are shown in Table 1 . In the table, each plan is set up to accomplish the higher hierarchy of the objective, leading to the completion of the research goal.

Table 1 Research Goal, Objective, and Plan

Goal	Objective	Plan
Develop a prototype which automatically decides construction WP from construction drawings according to standardized criteria	Objective 1: Organize detail section drawing data according to the RSMeans Assemblies Cost criteria	Plan 1: Collect and modify detail section drawings data
		Plan 2: Analyze the characteristic of drawings and text data
	Objective 2: Build up the structure of the automated WP decision prototype	Plan 3: Construct the text extraction model
		Plan 4: Construct the text classification model
	Objective 3: Evaluate the model performance to validate if the prototype produces the significantly accurate decision	Plan 5: Train and test the model
		Plan 6: Analyze the WP decision prototype evaluation plot

Original Contribution

This section explains the expected benefit within and outside of the Architecture, Engineering, and Construction (AEC) fields and presents the beneficiaries of this study.

This research is expected to assist in constructing WP and WBS without having broad CEM experience and knowledge. In addition, it will be helpful to establish a consistent WP format and convention among user groups. It is also expected to reduce confusion among project entities and increase WBS compatibility and tools involved in each project phase. This provides leeway to prospective BIM integration and other Smart

Construction technology. Moreover, this will significantly reduce the time and error of WBS arrangements compared to that of humans alone.

Not only can this research be utilized in the AEC field, but it also can be applied to other fields of study. This is possible by utilizing different datasets. For example, when medical documents are fed into the system, this will provide baseline output for automatic symptom analysis and the prescription assistant system. In addition, if sets of journal articles are involved, the output can significantly contribute to developing an automatic journal category classification system or provide a clue for characteristic analysis on a specific type of journal.

This research will benefit the assist project practitioner with a lack of Construction Engineering and Management (CEM) background by providing suitable WP options when the data is provided. Moreover, the beneficiaries of this research are not limited to the inexperienced. It will also benefit participants, including contractors, facility managers, architects, engineers, and project managers. This is because it helps provide the standardized format of WP, promoting compatibilities among each participants' work compositions, time-saving, and error reduction.

Key Assumption, Delimitation, Limitation

This section describes the key assumptions, delimitations, and limitations of this research.

Key assumptions refer to the statement or bias made prior to the research. First, material texts and their combinations among those texts represent the building component or WP. Second, building component type, WP, and data label are considered the same concept. Third, Machine Learning (ML) is a sub-concept of Artificial Intelligence (AI), while Deep Learning (DL) belongs to the ML category. Forth, the classification term is considered the WP decision process in this research. Fifth, abbreviation terms used in the detail section drawing are fully extended, except for the unit. Lastly, loss, accuracy, and evaluation matrix value well represent the performance of the prototype.

Delimitation is described to set up the boundary of this research. First, only text data in detail section drawing is used in research data. Although detail section drawing interpretation involves both reading texts and images, texts are mainly recognized for analysis of WP, whereas images are used as reference. For this reason, only texts (excluding numbers) are extracted and used to feed the prototype. Second, only one pdf file displays one building component each. This is because if the drawings contain multiple building components, this is expected to engender confusion for prototype training and testing. Third, as this research examines the viability of AI usage in WP, the following conditions are limited; 1. Detail section drawings which correspond to 3 divisions in RSMeans Assemblies Costs data (Gordian, 2017)(A1010 Standard Foundation, B2010 Exterior Wall, and C1010 Partition) are collected. 2. Detail section drawings from only commercial projects are used.

The limitation concerns the influence of the study, which the researcher cannot control. The DL process involves random data transmission within the neural network in this research. This means that even the same conditions are provided, the specific result value might vary depending on the specification of the computer hardware or software system.

CHAPTER II

LITERATURE REVIEW

Work Breakdown Structure and Work Packages

A guide to the Project Management Body of Knowledge (PMBOK), a publication of Project Management Institute (PMI), provides the definition and structure of WBS and WP. The example of WBS and WP hierarchy is shown in Figure 1. Work Breakdown Structure, or WBS, is a deliverable-based work hierarchy disintegration which is designed to be initiated by project team for project completion and construct project deliverable (PMI, 2017). The whole project activity, resources, and other information are defined within WBS, with the various WP integrated.

Besides, WP is a smaller part of WBS, enabling project management to decide the necessary steps for WP completion. Each WP consists of the necessary work involved and the deadline of the WP. It allows the systematic operation of different works by multiple parties. This is because it provides each team a guideline defined in WP work and the completion deadline.

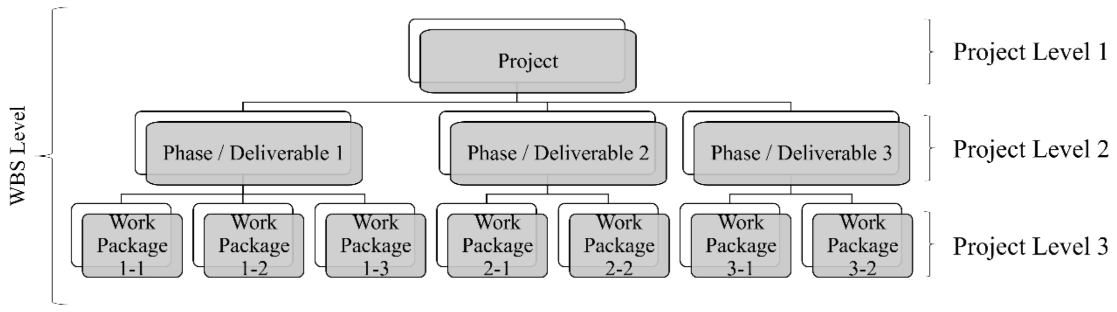
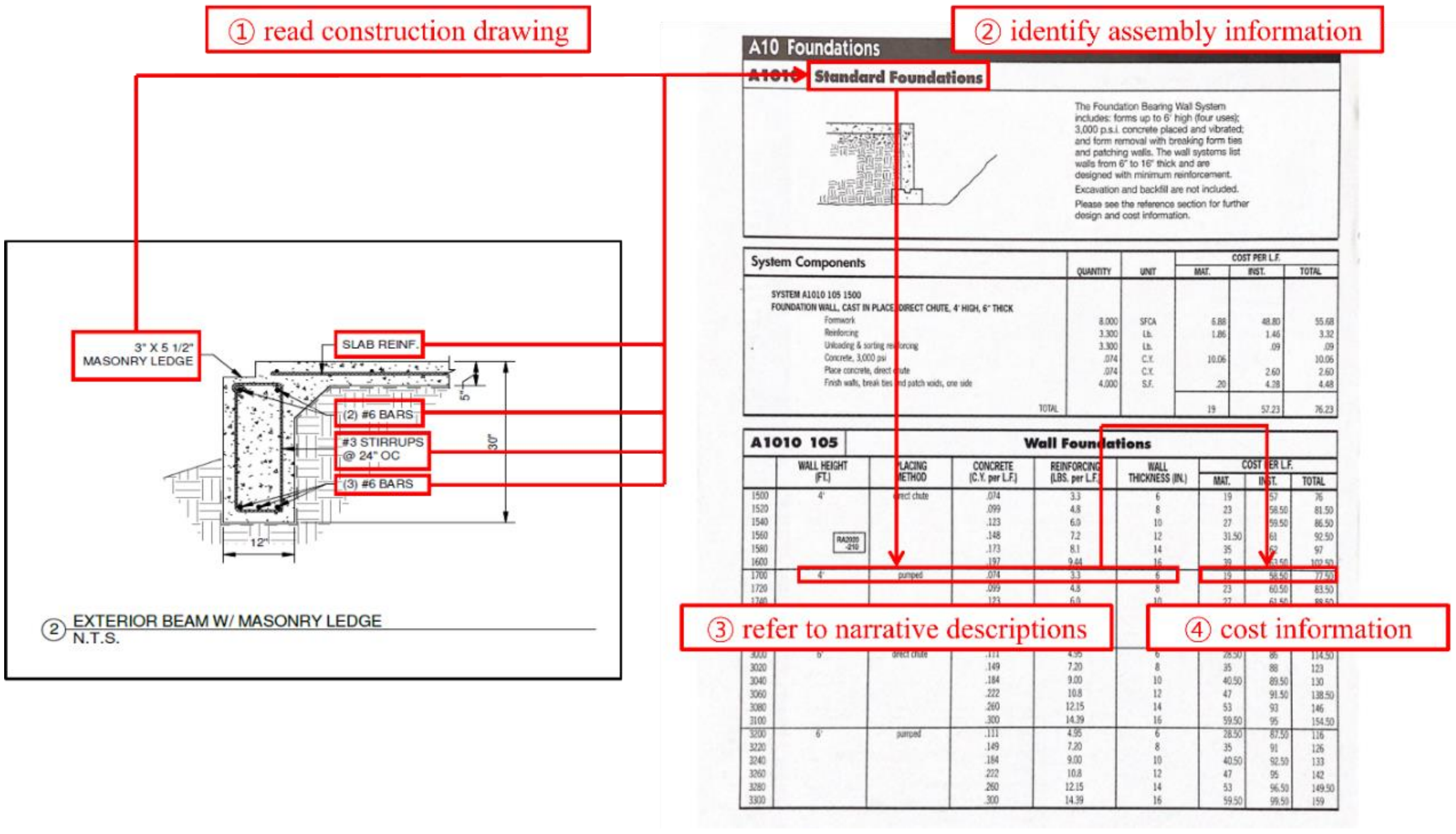


Figure 1 Example Structure of WBS and WP

RSMeans Assemblies Cost Data

RSMeans Assemblies Costs data is a construction cost information database for entities using the newest localized construction cost. This data is helpful because it helps identify and quantify new building products and methodologies and adjust productivity rates and costs to local market conditions(Gordian, 2017). It is mainly used in the early design stage by developing various design scenarios and comparing the cost impact of each alternative. This data is organized according to the American Society for Testing and Material (ASTM) UNIFORMAT II standard, a format used for classifying building components and associated site work (Charette & Marshall, 1999). The users read construction drawings, identify assembly information, refer to narrative descriptions to use cost information from the Assemblies Cost data. This procedure is shown in Figure 2.


① read construction drawing
② identify assembly information
③ refer to narrative descriptions
④ cost information
3" X 5 1/2" MASONRY LEDGE
SLAB REINF.
(2) #6 BARS
#3 STIRRUPS @ 24" OC
(3) #6 BARS
5"
30"
12"
2 EXTERIOR BEAM W/ MASONRY LEDGE
N.T.S.

A10 Foundations

A1010 Standard Foundations

The Foundation Bearing Wall System includes: forms up to 6' high (four uses); 3,000 p.s.i. concrete placed and vibrated; and form removal with breaking form ties and patching walls. The wall systems list walls from 6" to 16" thick and are designed with minimum reinforcement.

Excavation and backfill are not included.

Please see the reference section for further design and cost information.

System Components	QUANTITY	UNIT	COST PER L.F.		
			MAT.	INST.	TOTAL
SYSTEM A1010 105 1500 FOUNDATION WALL, CAST IN PLACE DIRECT CHUTE, 4' HIGH, 6" THICK					
Formwork	8.000	SFCA	6.88	48.80	55.68
Reinforcing	3.300	Lb.	1.86	1.46	3.32
Unloading & sorting reinforcing	3.300	Lb.		.09	.09
Concrete, 3,000 psi	.074	C.Y.	10.06		10.06
Place concrete, direct chute	.074	C.Y.		2.60	2.60
Finish walls, break ties and patch voids, one side	4.000	S.F.	.20	4.28	4.48
TOTAL			19	57.23	76.23

A1010 105 Wall Foundations

	WALL HEIGHT (FT.)	PLACING METHOD	CONCRETE (C.Y. per L.F.)	REINFORCING (LBS. per L.F.)	WALL THICKNESS (IN.)	COST PER L.F.		
						MAT.	INST.	TOTAL
1500	4'	direct chute	.074	3.3	6	19	57	76
1520			.099	4.8	8	23	58.50	81.50
1540			.123	6.0	10	27	59.50	86.50
1560	RA2020 -210		.148	7.2	12	31.50	61	92.50
1580			.173	8.1	14	35	62	97
1600			.197	9.44	16	39	63.50	102.50
1700	4'	pumped	.074	3.3	6	19	58.50	77.50
1720			.099	4.8	8	23	60.50	83.50
1740			.123	6.0	10	27	61.50	88.50
3000	6'	direct chute	.111	4.95	6	28.50	86	114.50
3020			.149	7.20	8	35	88	123
3040			.184	9.00	10	40.50	89.50	130
3060			.222	10.8	12	47	91.50	138.50
3080			.260	12.15	14	53	93	146
3100			.300	14.39	16	59.50	95	154.50
3200	6'	pumped	.111	4.95	6	28.50	87.50	116
3220			.149	7.20	8	35	91	126
3240			.184	9.00	10	40.50	92.50	133
3260			.222	10.8	12	47	95	142
3280			.260	12.15	14	53	96.50	149.50
3300			.300	14.39	16	59.50	99.50	159

Figure 2 RSMeans Assemblies Costs Data Utilization Procedure Modified from Gordian (2017)

Detail Section Drawing

A detail section drawing is a portion of a building drawn at a large scale to clarify the construction assembly requirement (Brown & Dorfmueller, 2019). Reading and interpreting detail section drawing is vital when using the RSMeans Assemblies data. It is because the detail section drawing contains text format information such as dimensions, notation, materials, and others, as shown in Figure 3. By reading and interpreting text from detail section drawings, readers can understand which building component complies with the construction assembly requirements.

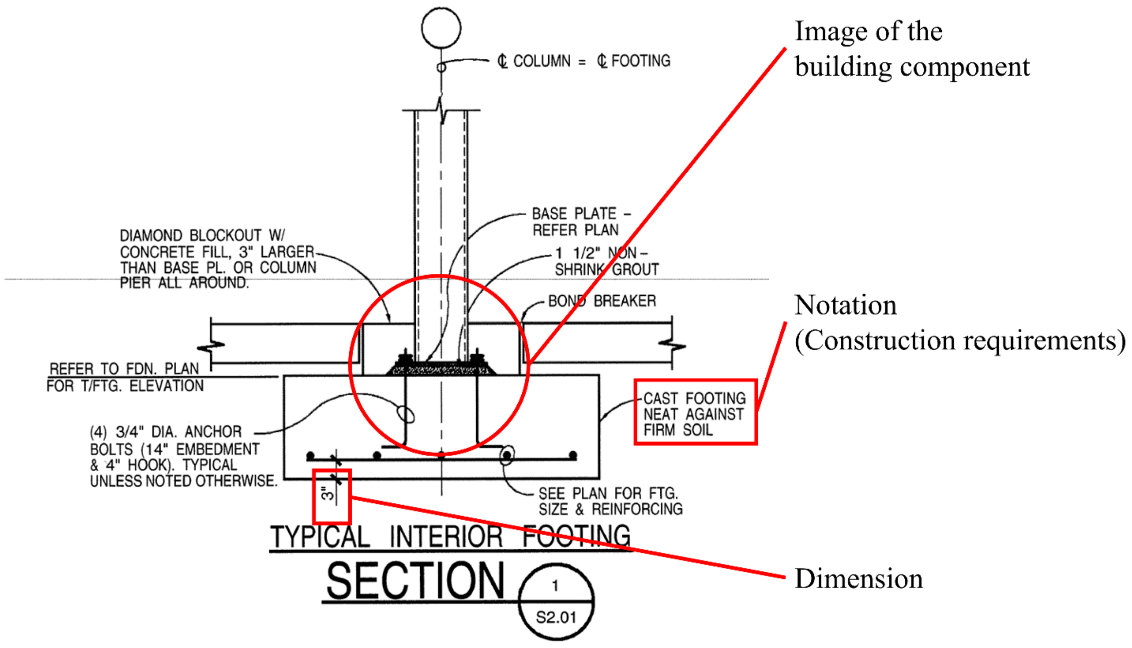

℄ COLUMN = ℄ FOOTING
Image of the building component
BASE PLATE – REFER PLAN
DIAMOND BLOCKOUT W/ CONCRETE FILL, 3" LARGER THAN BASE PL. OR COLUMN PIER ALL AROUND.
1 1/2" NON-SHRINK GROUT
BOND BREAKER
Notation (Construction requirements)
REFER TO FDN. PLAN FOR T/FTG. ELEVATION
CAST FOOTING NEAT AGAINST FIRM SOIL
(4) 3/4" DIA. ANCHOR BOLTS (14" EMBEDMENT & 4" HOOK). TYPICAL UNLESS NOTED OTHERWISE.
3"
SEE PLAN FOR FTG. SIZE & REINFORCING
TYPICAL INTERIOR FOOTING
SECTION
1
S2.01
Dimension

Figure 3 Composition of Detail Section Drawing Modified from Brown and Dorfmueller (2019)

AI-based Automation in AEC field

Industry 4.0 refers to the industry's next step, which focuses on bringing in manufacturing environment flexibility, which leads to improved productivity and customized products with enhanced quality (Kumar et al., 2019). In the era of Industry 4.0, the efficiency of the overall construction project is increasing with automation. Automation refers to the operation, action, or self-regulation which does not involve manual intervention (Nof, 2009). Especially, Artificial Intelligence (AI) promotes constant innovation of CEM, realizing a considerable boost in project automation (Pan & Zhang, 2021). It is because AI insights assist managers in understanding the construction project better, formalizing tacit knowledge from project experiences, and rapidly spotting the project concerns in a data-driven manner (Hu & Castro-Lacouture, 2019). Due to these benefits, many construction firms invest, adopt, and use AI in construction projects (Bughin et al., 2017). Figure 4 shows the proportion of each sector among AI adopting companies. In this chart, the construction sector is extruded to highlight the proportion of the construction sector.

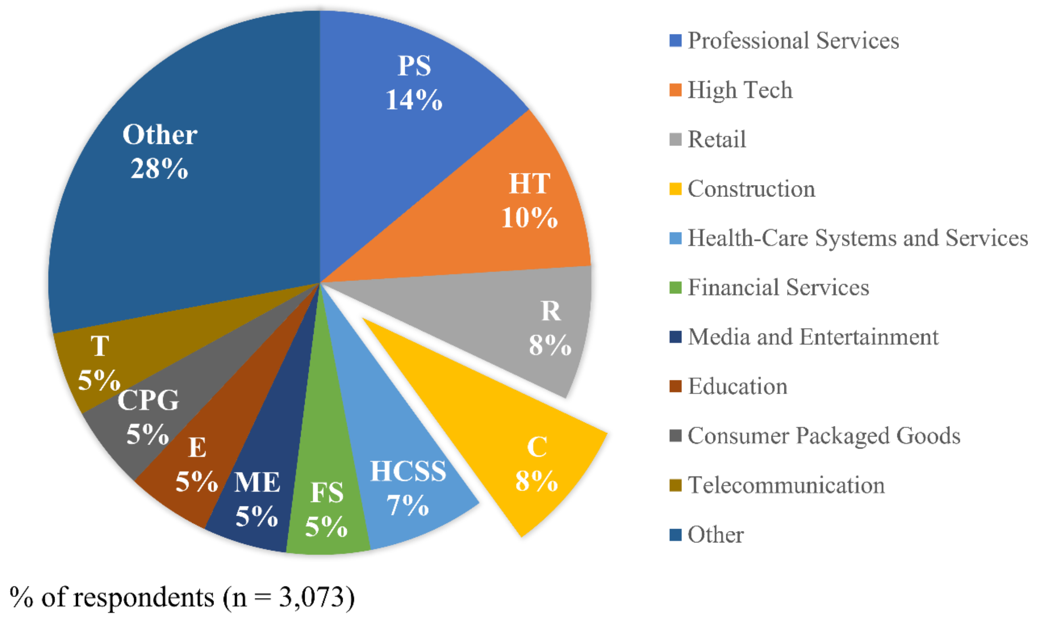


Figure 4 AI Adoption and Use Survey by Sector Adapted from Bughin et al. (2017)

As the construction industry is undergoing such a transition, there have been continuous attempts to apply AI techniques to AEC-related works.

Huang and Zheng (2018) use the Generative Adversarial Network (GAN) to recognize and generate architectural drawings, marking rooms with different colors and then generating apartment plans through Convolutional Neural Network (CNN). GAN, one type of ML works by providing training data in pairs. With the pair data, it discovers the most suitable parameter from the network, which makes the generated data similar to that of the original data. In this research, a pix2pixHD, one type of GAN, is used to learn floor plan data in pairs and creates a new floor plan based on the input.

Mewada et al. (2020) propose a floor plan information retrieval algorithm that can extract and recognize texts, symbols, or graphics to retrieve the floor plan information from the image. This algorithm is divided into two parts: Shape extraction and Room classification. First, textual information (text, label, symbol) and graphical information (topological shape and connectivity) are extracted from the floor plan. After extraction, the algorithm classifies the shape as a room or non-room type depending on the extracted parameters of text and graphics. When identifying a room type, the algorithm utilizes a logistic regression model, a type of ML. The core of the logistic regression model is a sigmoid function, shown in Figure 5. In the equation shown in Equation 1, x is a vectorized input data (room area and aspect ratio). When the input vector is presented into the function, the function gives out a probability value between 0 and 1. If the value exceeds 0.5, it is considered the default output type (room-type), and otherwise, non-room type.

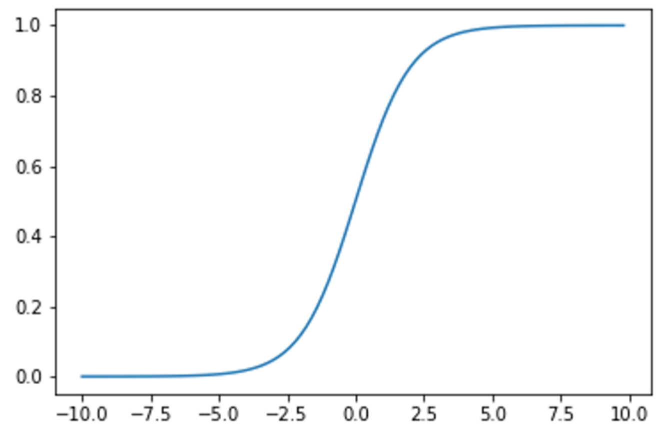


Figure 5 Sigmoid Graph

$$P = \frac{1}{1 + e^{-x}}$$

Equation 1 Sigmoid Function

Rho et al. (2020) suggest a framework to generate an object-oriented BIM model using drawing recognition and line-text extraction techniques. The text data located next to a polygon are extracted with text/graphic separation methods. The author introduces a Bayesian filter to identify the building component modeled from extracted text data. Using the Bayesian filter, the author developed a text classifier. This classification model determines which elements are presented according to the specific abbreviations from the drawing. This determination is possible after training abbreviations from the structural list and the structural element's actual name.

DL Based Text Classification Model

AI can assist in identifying the assembly identification process from detail section drawings done by human activity. With AI, the computer can automatically process, investigate, and interpret texts and produce a meaningful result similar to human understanding of natural language (Pan & Zhang, 2021). This approach is called Natural Language Processing (NLP).

One of the applications of NLP is classification. NLP is used to identify higher-hierarchy information from text data (Wu et al., 2020). Detail section drawing contains representative text data (dimensions, notations, symbols, and others). With these data, the NLP program can exploit these text data to identify the higher class of assembly information attained from the drawing, such as building components, construction requirements, and others.

Above various AI techniques, DL is considered a robust tool for constructing an NLP model. DL is a method which uses a computation model inspired by how the human brain processes data or neural networks (Kelleher, 2019). The neural networks consist of units called neurons. Each neuron processes the given information from neighboring layer, decide to transmit either high value or low value signal. These neurons are mutually connected to process information and spread throughout the network, as shown in Figure 6. In this figure, small square and circle refers to neurons, while the arrow represents the network. The reason for its significance is that it utilizes a large amount of data to acquire the contextual meaning of the word from complicated sentence structure (Zhou, 2022). This leads to better model performance compared to other conventional ML methods.

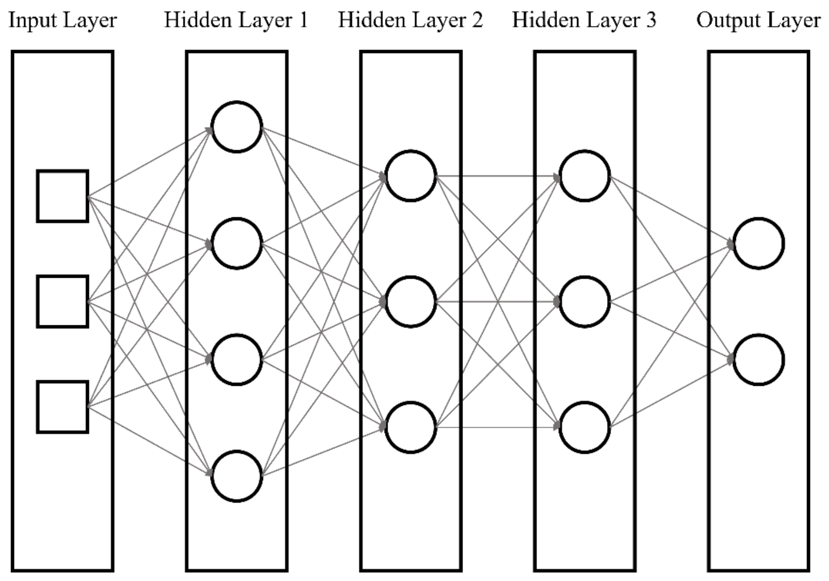


Figure 6 Sample Neural Network Adapted from Kelleher (2019)

There are two main DL techniques: Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). The structure and application examples are as follows.

CNN refers to a structure where the front part of layers extracts the local features of the data and integrates those into the subsequent layers, forming a hierarchy among those features (Kelleher, 2019). As an example for this structure, Ce and Tie (2020) present an analysis method for interpretability of a CNN text classification model is proposed. The paper mentions that CNN is efficient since it can achieve better prediction accuracy and utilizes fewer computing resources than other ML methods. The paper also presents the CNN- based model structure, which is shown in Figure **7**. Text data is processed by segmenting words and removing stop words. Then, the words are transformed into numerical vector values (word embedding). Those vectors are stored in V (vocabulary size) * D (dimension of word embedding) sized layer called embedding layer. After storing data, the data goes through the convolutional layer, pooling layer, and connection layer. This structure is called CNN. This structure uses a method called max-pooling. The max-pooling method is used because it extracts the maximum value in each feature graph by the feature graph unit, which is necessary for global maximum pooling on the classification layer. Then the data goes through the classification layer for the classification output.

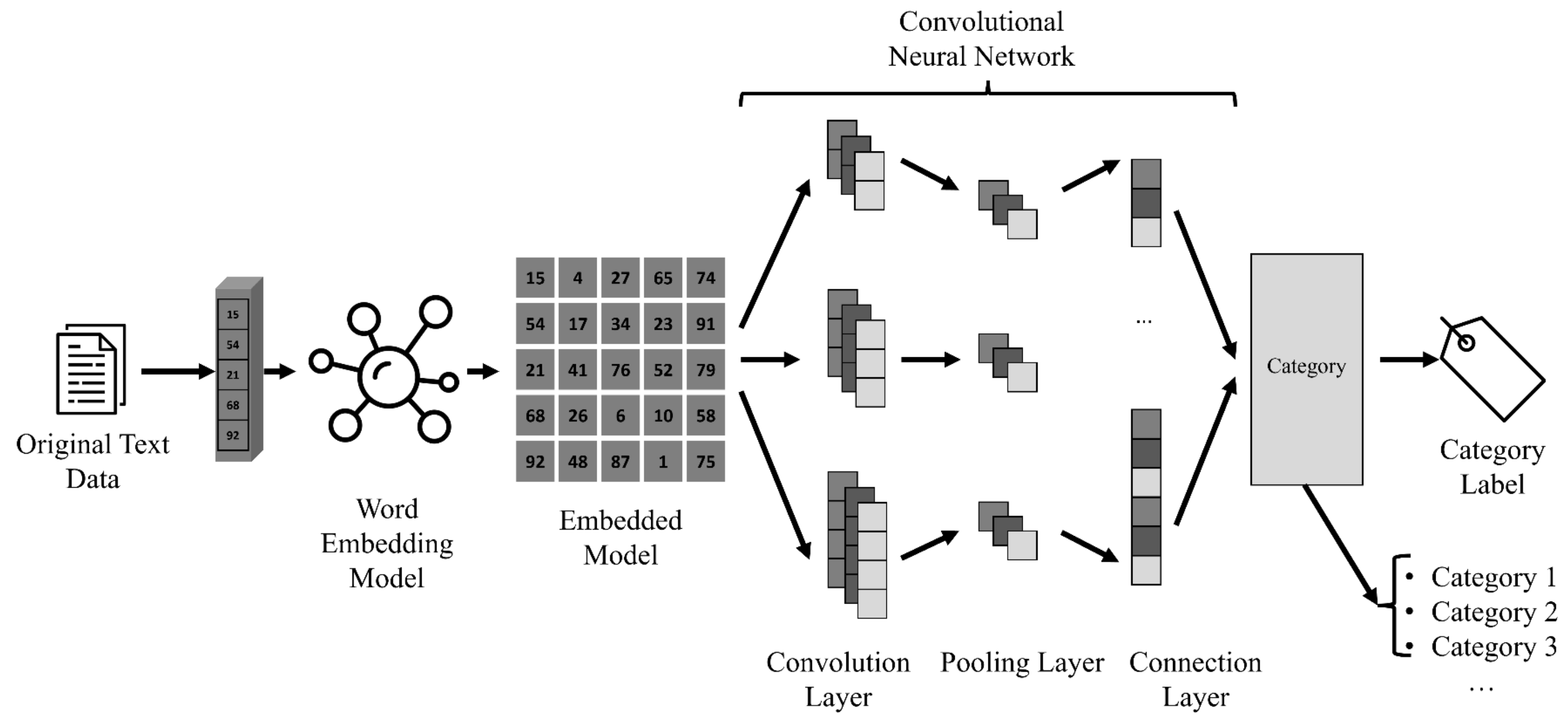
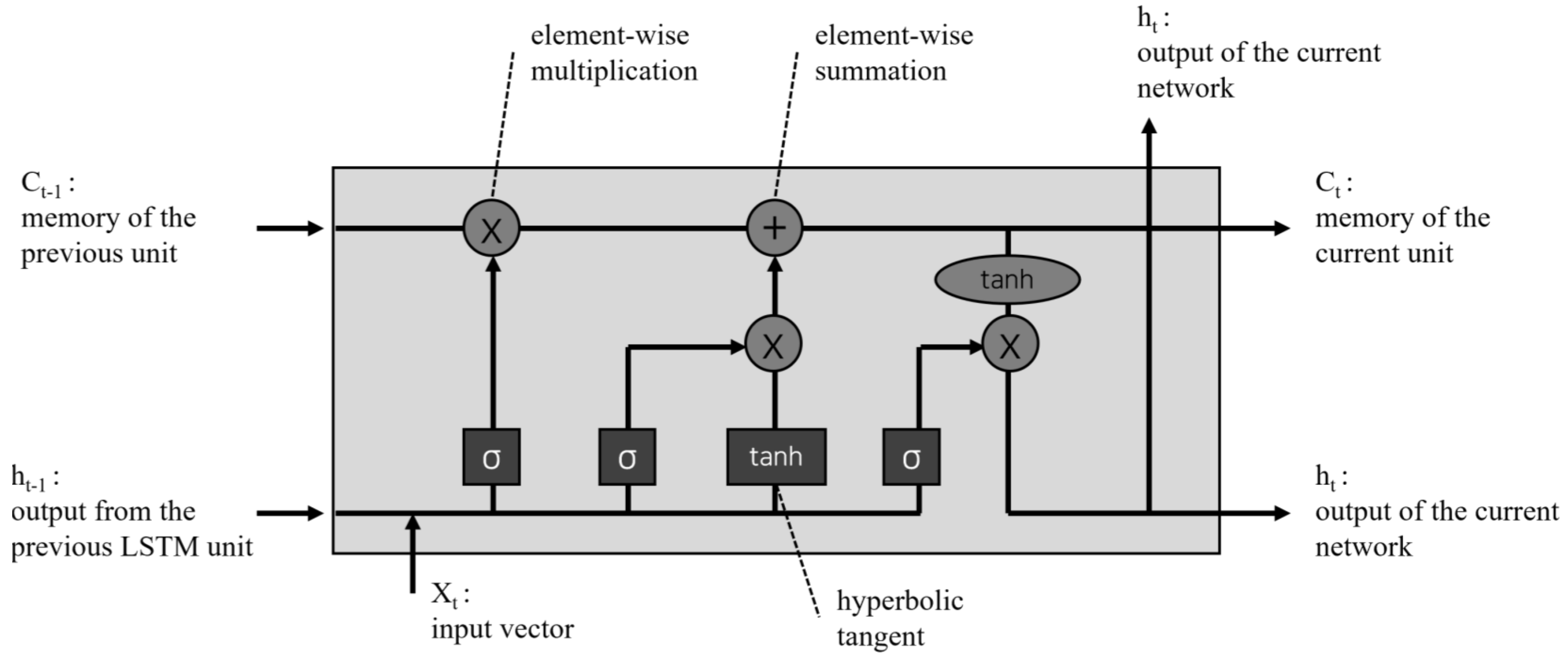


Figure 7 Structure of CNN Modified from Ce and Tie (2020)

RNN is a type of DL model which use context information in order to map sequential data between input and output (Graves, 2012). As an example of RNN text classification, Che-Wen et al. (2021) proposed an outpatient classification model using the Long-Short Term Memory (LSTM) model, one type of RNN. In this study, the model is designed to identify outpatient categories according to the text data. The text data contains the user’s inquiry about the disease and the doctor’s professional answers. The author uses Long-Short Term Memory, for the classification model. The major problem of previous RNN is that the significance of hidden layers between input and output gradually weakens as the input data pass through the network. This phenomenon is called vanishing gradient problem (Bridle, 1990). As a solution, the concept of the LSTM is introduced. The overall structure of LSTM is shown in Figure 8. Unlike other ML models, this modulates the memory of each learning step. This becomes possible because of the functions inside the LSTM neuron cell. Those functions in LSTM cell decides whether to memorize or forget the data from input vector, and transfer as an output vector. This method is beneficial when integrating long-term dependency among each training step and finding features in the sequence of sentences. In the evaluating steps, the author compares accuracy, precision, recall, F1-score values with other machine learning models (Naïve Bayes, K-Nearest Neighbor, Supporting Vector Machine, CNN, FastText, Transformer). Among these models, the author concludes that LSTM shows the highest performance and accuracy.



C_{t-1} :
memory of the previous unit
h_{t-1} :
output from the previous LSTM unit
X_t :
input vector
element-wise multiplication
element-wise summation
hyperbolic tangent
h_t :
output of the current network
C_t :
memory of the current unit
h_t :
output of the current network
X
+
σ
tanh

Figure 8 Structure of Long-Short Term Memory (LSTM) Adapted from Che-Wen et al. (2021)

Zhou et al. (2015) combine the strength of CNN and RNN architectures and propose a novel and unified model called C-LSTM for sentence representation and text classification. CNN specializes in learning local responses from the individual words but lacks the ability to learn the sequential correlation between words. On the other hand, RNN performs well in words with sequence. This study uses CNN to extract a higher-hierarchy word sequence of target words and then obtain the sentence representation with RNN methods.

Validation Set Method

James et al. (2021) demonstrate how model significance should be validated and several validation methods. In order to examine the viability of the study, the prototype needs to generate results with significantly low test error. Test error refers to the average error from prediction output on observations not used for the training process. If training data and testing data overlaps, the training error compromises the result of testing data. Although, the testing error can be calculated when a designated test set is given. However, the large amount of designated test datasets is usually unavailable.

One of the methods to overcome this shortage issue is the validation set method. This associates the data set, which is randomly divided into training and validation sets. Training data fits the model in each random extraction scenario, whereas the validation set is adopted for the prediction. After the validation set is used, the model which produced the lowest test error is picked. The example of the validation set method is shown in Figure 9.

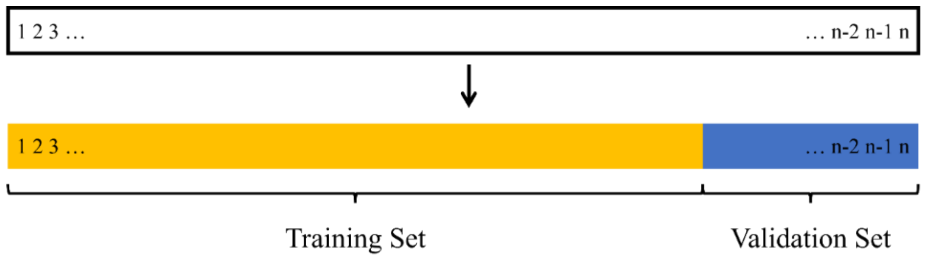


Figure 9 Example of Validation Set Method Adapted from James et al. (2021)

Multi Class Evaluation Matrix

Evaluation measures are used to evaluate how well the classification model performs (Khan et al., 2022). Among various metrics, example-based metrics acquire an evaluation from every data and calculate the average among the entire dataset (Krstinic et al., 2020). Let N be the number of given data. When Y_i is defined as ground truth label if data x_i is given, while Z_i is defined as the prediction of the label given x_i, example-based metrics are shown below. Equation 2 to Equation 5 represents accuracy, precision, recall, and F-1 score.

$$Accuracy = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Y_i \cup Z_i|}$$ **Equation 2 Example – Based Accuracy**

$$Precision = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Z_i|}$$ **Equation 3 Example – Based Precision**

$$Recall = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i \cap Z_i|}{|Y_i|}$$ **Equation 4 Example – Based Recall**

$$F - 1\,Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

$$= \frac{1}{N}\sum_{i=1}^{N}\frac{2|Y_i \cap Z_i|}{|Y_i| + |Z_i|}$$ **Equation 5 Example – Based F-1 Score**

This evaluation method was inspired by the work of Che-Wen et al. (2021). As shown in Table 2, this research compares evaluation matrix values of the LSTM based model with other type of ML model. This studied mentioned that proposed LSTM based model performed 95% accuracy, precise, recall, and 94% on F-1 score on eight-class text classification, concluding that this model produces significantly accurate classification result.

Table 2 Comparison of the Different Methods on Eight-Class Classification Modified from Che-Wen et al. (2021)

Method	Accuracy	Precise	Recall	F-1 Score
NB	90%	91%	90%	89%
KNN	64%	78%	64%	65%
SVM	90%	91%	90%	90%
CNN	93%	94%	94%	93%
FastText	93%	94%	94%	93%
Transformer	94%	94%	94%	94%
Proposed (LSTM)	95%	95%	95%	94%

.

Conclusion

This research aims to construct a prototype which conducts automated WP decisions from detail section drawing. In order to establish standardized criteria for WP, RSMeans Assemblies Costs data is chosen. This is because it follows the UNIFORMAT II standard, which systematically divides the overall project into workable building components.

AEC filed the reason for the increasing application of AI-based automation: assisting the better understanding of project management, organizing implicit project experience, and accelerated data-driven solutions on the project obstacle. Among various AI technologies, DL is the most suitable method for NLP model development since it is capable of processing large amounts of contextual text data. CNN and RNN are representative methods of the DL method. RNN, especially the LSTM method, will be adopted in this research. The selection depends on the characteristics of input and output data. CNN is selected if the independent representation of individual words is the target output. On the other hand, RNN(LSTM) is used when the textual correlation is the desired output. As the target output in this research is based on the relationship among texts in this project, the RNN model is chosen.

The validation set method will be utilized in this project when running the model. Moreover, for evaluating the performance of the prototype, an evaluation matrix will be used.

CHAPTER III

RESEARCH METHODOLOGY

This chapter demonstrates methods to accomplish each plan for object completion. Table 3 presents the hierarchy of research plan, methods, involved data, and method deliverable. In addition, Figure 10 shows how this will be delivered.

Table 3 Research Plan, Method, Data, and Deliverables

Plan	Method	Data	Deliverable
• Plan 1: Collect and modify detail section drawings data	• Manual collection of drawing • Manual modification of the drawing with AutoCAD	• 35 detail section drawings or schedules from 12 commercial construction projects • 11 RSMeans Assemblies Costs Sets	• To collect data and preprocess data which will be used to fit the prototype
• Plan 2: Analyze the characteristic of drawings and text data	• Categorize drawings according to RSMeans Assemblies Costs data • Count the word frequency of text data within each label	• RSMeans Assemblies Costs data • CSV file containing drawing texts and label	• To understand the characteristics of the data
• Plan 3: Construct the text extraction model	• Compose pdfminer module-based model	• Detail section drawing pdf file of each label	• To create the model to extract data needed for the text classification model automatically

Table 2 Continued

Plan	Method	Data	Deliverable
• Plan 4: Construct the text classification model	• Compose preprocessing, LSTM based module	• CSV file containing drawing texts and label	• To create the model which automatically decides WP type
• Plan 5: Train and test the model	• Run the model and calculate training/validation loss and accuracy in each epoch, • Test the model with testing data	• Preprocessed text and label data, • Prediction result from the validation set	• To track whether the models perform in the desired way
Plan 6: Analyze the WP decision prototype evaluation plot	• Interpret training/validation loss and accuracy, and evaluation matrix values from testing result	• Training/validation loss and accuracy, and evaluation matrix values	• To validate the viability of the prototype

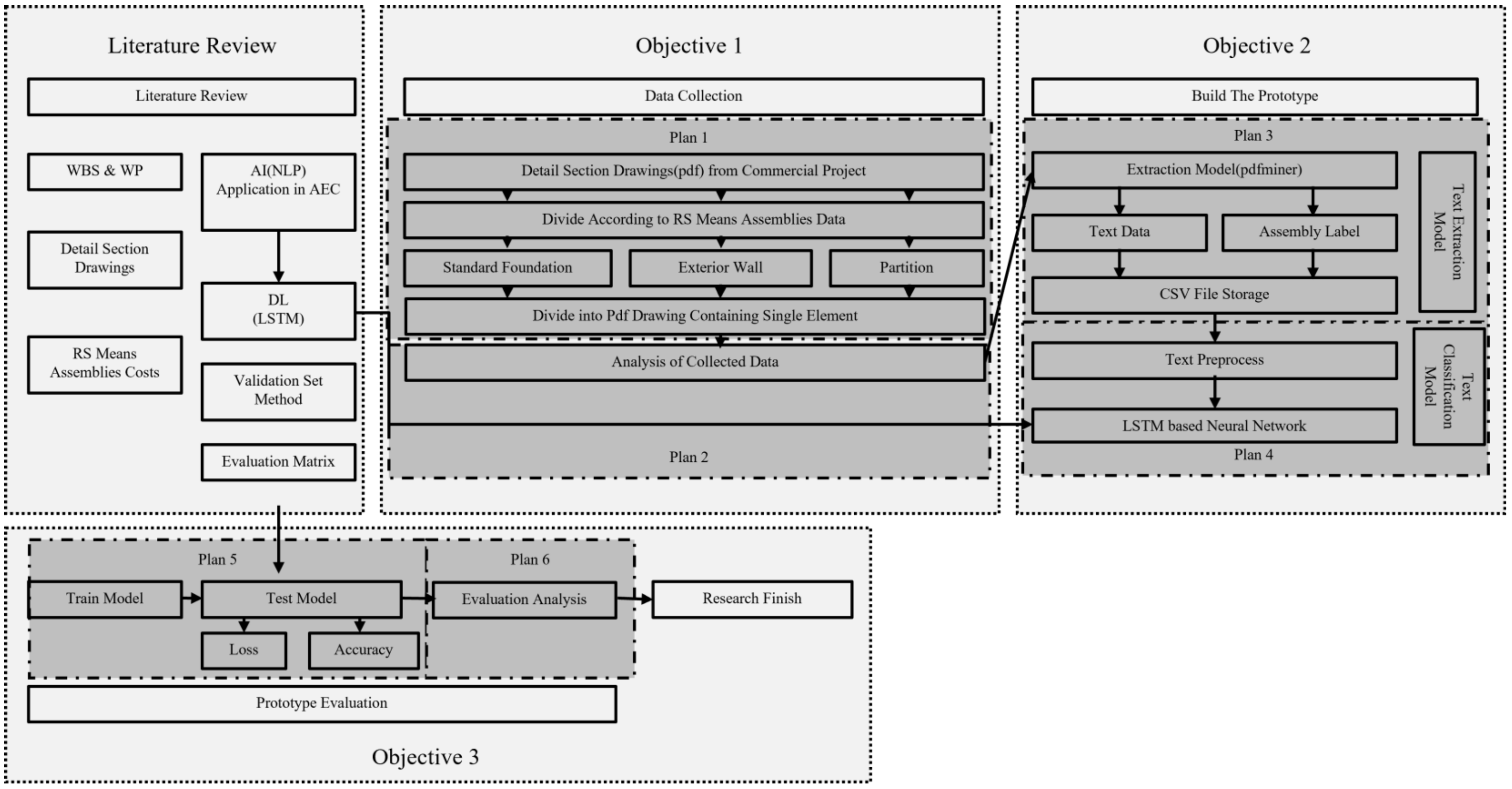


Figure 10 Workflow of the Research

Data Collection and Structure

This section presents how these data were collected and the data structure involved in this research.

Target data are detail section drawings sets which contain text data. The text data include material and annotation information representing the building element's composition. This project selected 35 detail section drawings or schedules from 12 commercial construction projects and 11 RSMeans assemblies information sets. These sets include the following building component type selected from the RSMeans assembly section: Standard foundation, exterior wall, and partition. An example of partition detail section drawings sets from Cibolo Fire Station No.2 architectural plan drawing is presented in Appendix B-1. The summary of the detail section drawings, RSMeans Assemblies data amount is shown in upper 2 rows of Table 4.

After collecting detail section drawings and schedule sets, those sets are divided into drawing data containing single components. This division process is done with AutoCAD. A drawing set is imported into AutoCAD. After importing, only text and images of a single component are exported into pdf format. Figure 11 shows the example of an exported single component of partition drawing, which is partition A type from the Cibolo Fire Station project. As a result, 102 standard foundations, 109 exterior walls, and 103 partition drawings are the output of this procedure. The summary of each component amount is shown in lower 3 rows of Table 4.

Table 4 Collected Data Summary

Data Type	Amount (ea.)
Total Detail Section Drawings or Schedule Sets	35
RSMeans Assemblies Costs Data	11
Standard Foundation Drawings Texts	102
Exterior Wall Drawings Texts	109
Partition	103

Texts containing material information from the drawings are extracted with the text extraction model. The structure of the text extraction model is presented in the following subchapter. As an output of the model, a CSV file with texts from a single component drawing and a label containing the building component type is created. CSV file containing 314 data with 'text' and 'label' column is created. Figure 12 is a partial CSV file containing texts and label data.

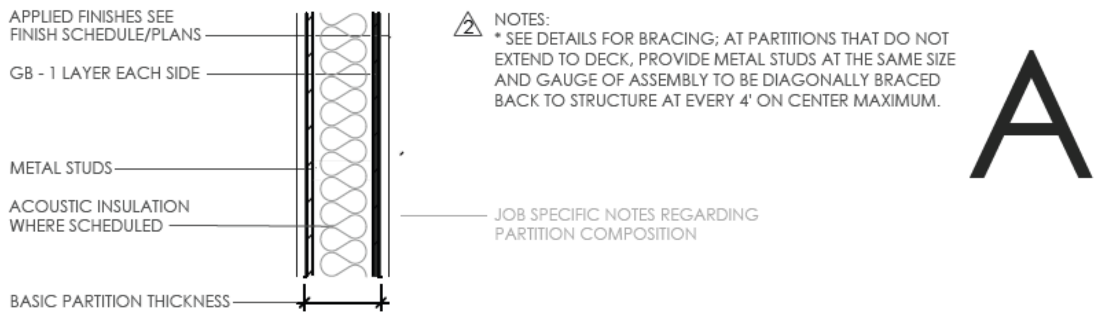
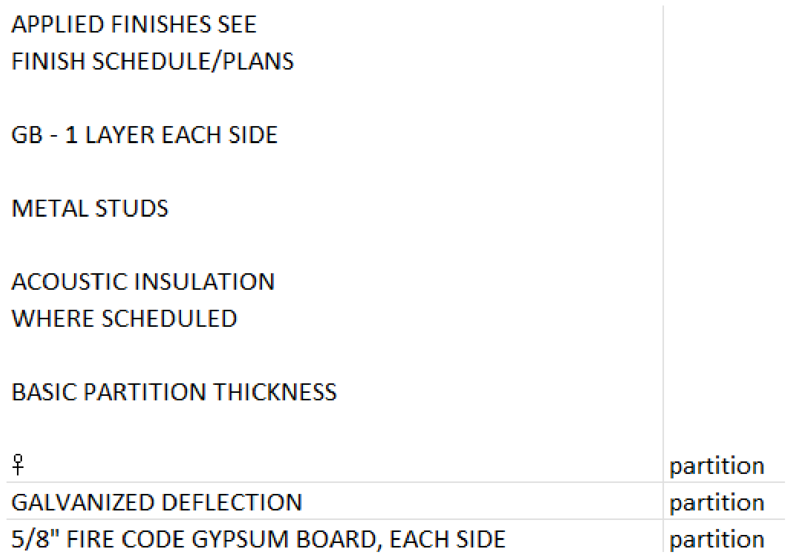

APPLIED FINISHES SEE FINISH SCHEDULE/PLANS
GB - 1 LAYER EACH SIDE
METAL STUDS
ACOUSTIC INSULATION WHERE SCHEDULED
BASIC PARTITION THICKNESS
2
NOTES:
* SEE DETAILS FOR BRACING; AT PARTITIONS THAT DO NOT EXTEND TO DECK, PROVIDE METAL STUDS AT THE SAME SIZE AND GAUGE OF ASSEMBLY TO BE DIAGONALLY BRACED BACK TO STRUCTURE AT EVERY 4' ON CENTER MAXIMUM.
JOB SPECIFIC NOTES REGARDING PARTITION COMPOSITION
A

Figure 11 Sample Partition Single Detail Section Drawing Modified from Simpson (2009)



APPLIED FINISHES SEE FINISH SCHEDULE/PLANS GB - 1 LAYER EACH SIDE METAL STUDS ACOUSTIC INSULATION WHERE SCHEDULED BASIC PARTITION THICKNESS ♀	partition
GALVANIZED DEFLECTION	partition
5/8" FIRE CODE GYPSUM BOARD, EACH SIDE	partition

Figure 12 Partial CSV File Containing Text and Label for Partition Sample

As the model highly relies upon word sequence when training and testing, each word token frequency is significant for the prediction. In other words, the more frequently a particular type of word token appears for given data, the more it will give weight for deciding on the building component type. Figure 13 presents the top 10 word frequency of the entire dataset. Specifically, Figure 14 to Figure 16 provide the top 10 word frequency of the data within the following labels: Standard foundation, Exterior wall, and partition. In those graphs, the height of each bar represents the frequency of each word token. The code for plotting the frequency bar chart is presented in Appendix A-1.

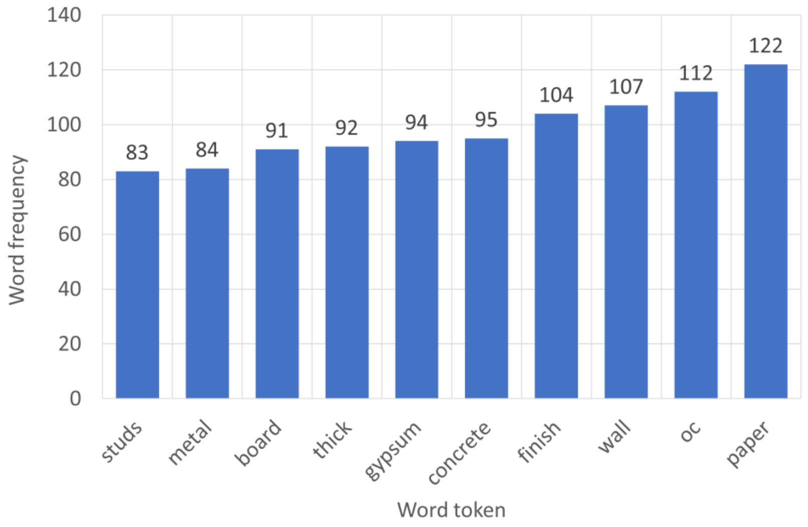


Figure 13 Top 10 Word Frequency of Overall Drawings

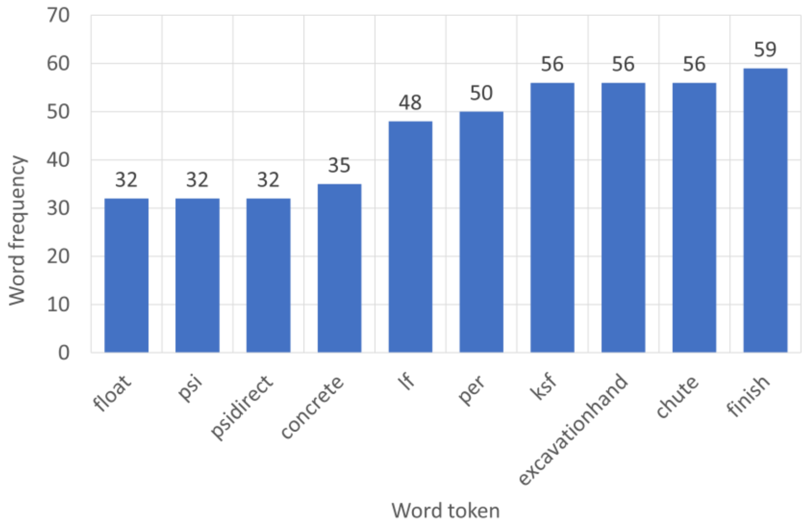


Figure 14 Top 10 Word Frequency of Standard Foundation Drawings

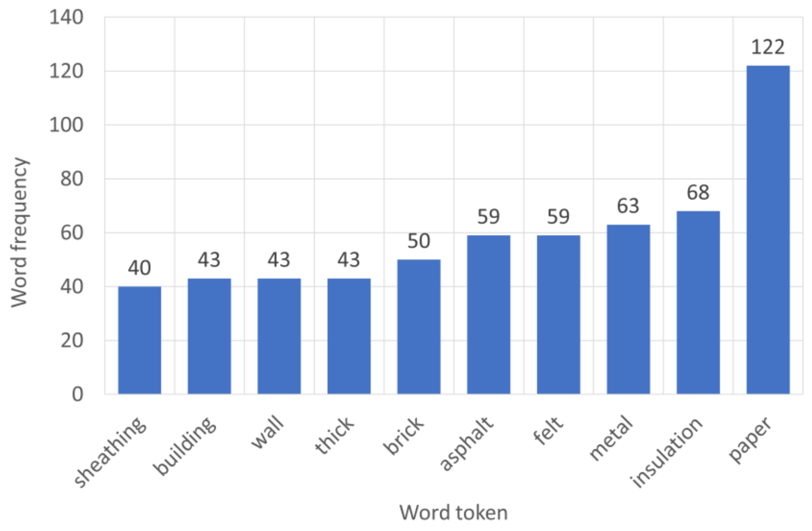


Figure 15 Top 10 Word Frequency of Exterior Wall Drawings

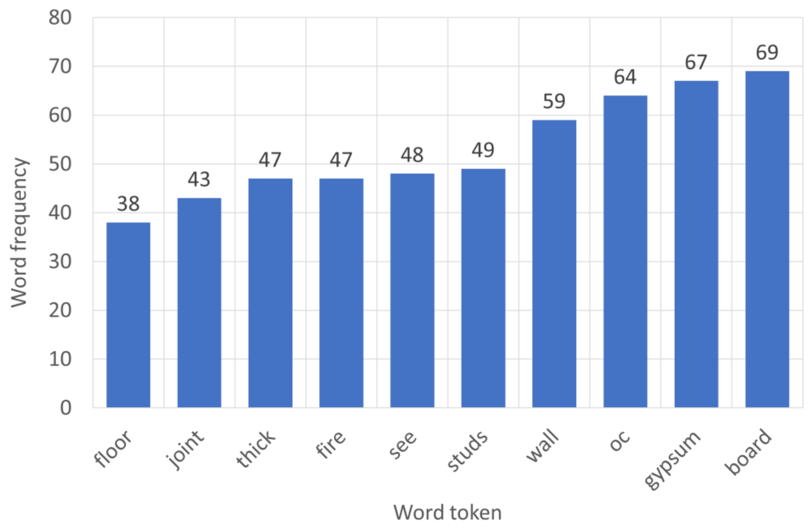


Figure 16 Top 10 Word Frequency of Partition Drawings

Automated WP Decision Prototype

This chapter demonstrates the overall structure and workflow of the automated WP decision prototype. The formation and data process of the prototype are shown in Figure 17.

The architecture of the prototype consists of two major parts: The text extraction model and the Text Classification Model. The text extraction model works as a virtual extractor that scans through the document and pulls out text. Meanwhile, the text classification module is a core part of this prototype which perceives text data and makes a decision just like the human WP decision procedure. Each model's detailed composition and subsequent structure are described in the following subchapters.

The overall workflow of the prototype contains two major data pipelines: Training data flow and Testing data flow. This pipeline penetrates the elements in the WP decision-making prototype in the following sequence. The detail section drawings in the pdf file are imported from the data source. After importing, the data undergoes a text extraction model, generating a CSV file with extracted texts from the drawings and the building property label. Next, the text classification model utilizes texts and labels from the CSV file. This dataset is used to train and validate the text classification model. After the text classification model is built with the training process, the user provides the new testing data. The prototype will use the data and decide which WP to utilize with texts from the new testing data. The specific data procedure and output are also explained in the following subchapters.

This structure was coded in the Google Colaboratory environment because it utilizes cloud storage, enabling users to easily access and use the program wherever the internet browser is available. The overall program was written with Python language as this language provides a flexible baseline to utilize various objects created by other programmers.

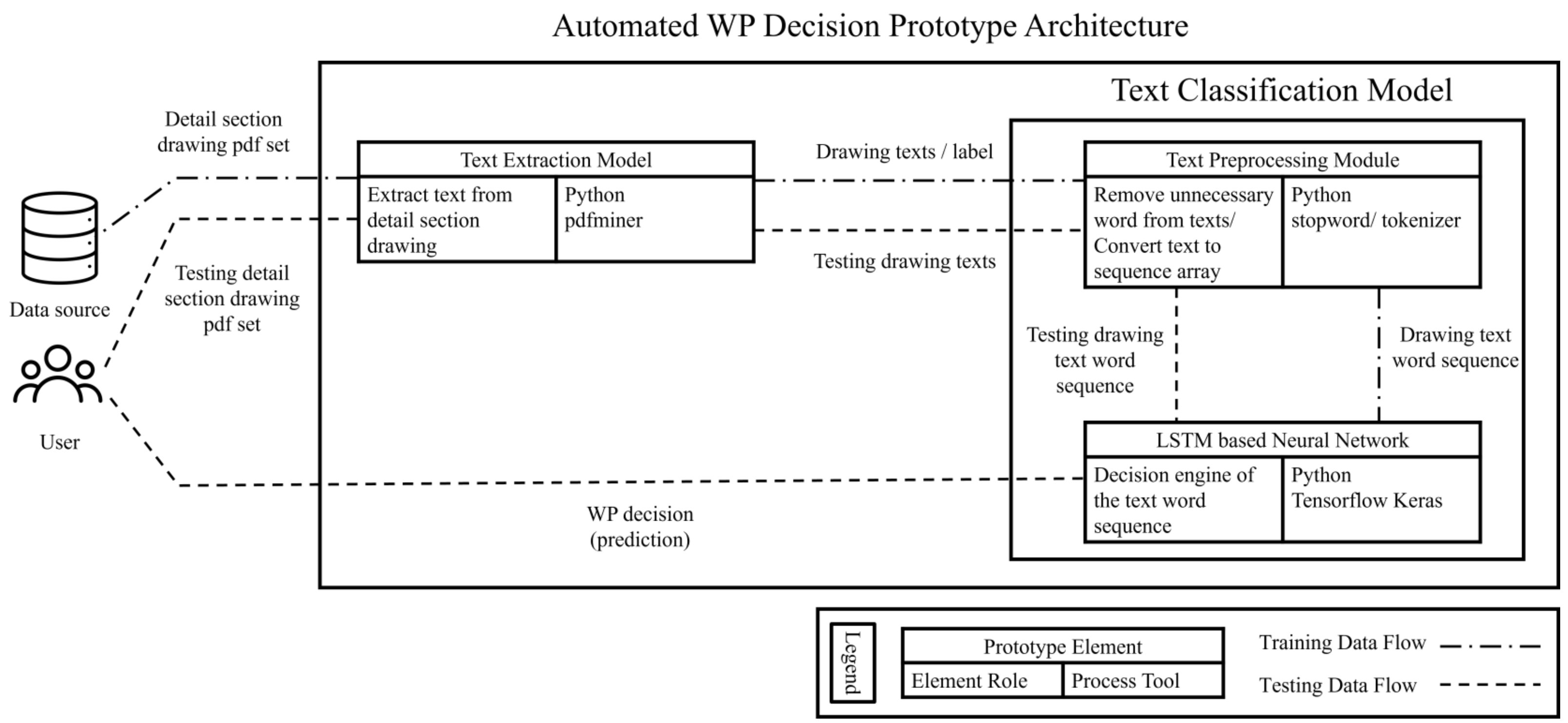

Automated WP Decision Prototype Architecture
Text Classification Model
Detail section drawing pdf set
Data source
Testing detail section drawing pdf set
User
Text Extraction Model
Extract text from detail section drawing
Python pdfminer
Drawing texts / label
Testing drawing texts
Text Preprocessing Module
Remove unnecessary word from texts/ Convert text to sequence array
Python stopword/ tokenizer
Testing drawing text word sequence
Drawing text word sequence
LSTM based Neural Network
Decision engine of the text word sequence
Python Tensorflow Keras
WP decision (prediction)
Legend
Prototype Element
Element Role
Process Tool
Training Data Flow
Testing Data Flow

Figure 17 Automated WP Decision Prototype Architecture and Workflow

Text Extraction Model

This section describes the structure and procedure of the text extraction model.

The overall composition of this model is shown in Figure 18. This model consists of data source, pdf extraction, and CSV writing. The data source contains a set of pdf files from the data collection process. Pdf extraction part, which utilizes high_level module from pdfminer library. This module scans through the entire pdf page and withdraws text data. Meanwhile, CSV writing creates a new CSV file and records the text and number from the input data.

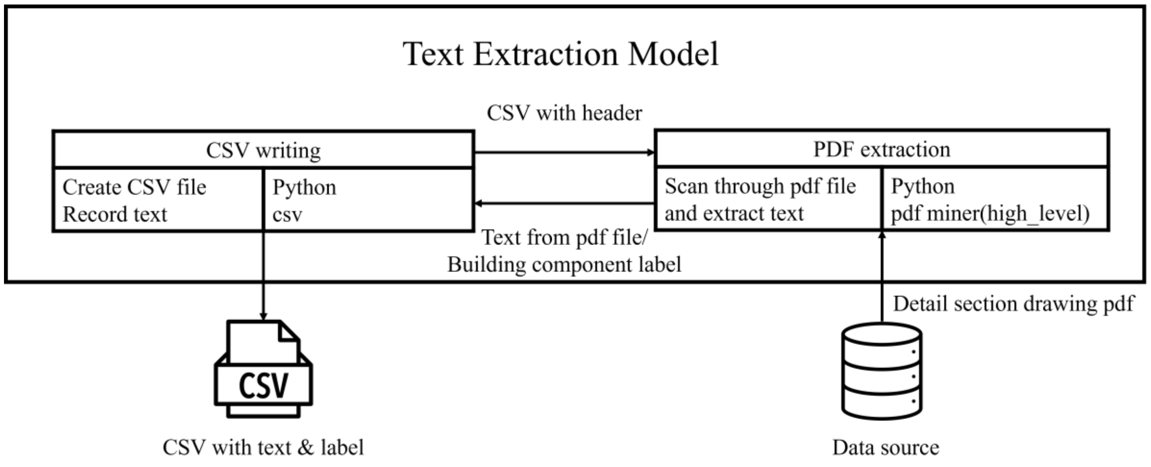
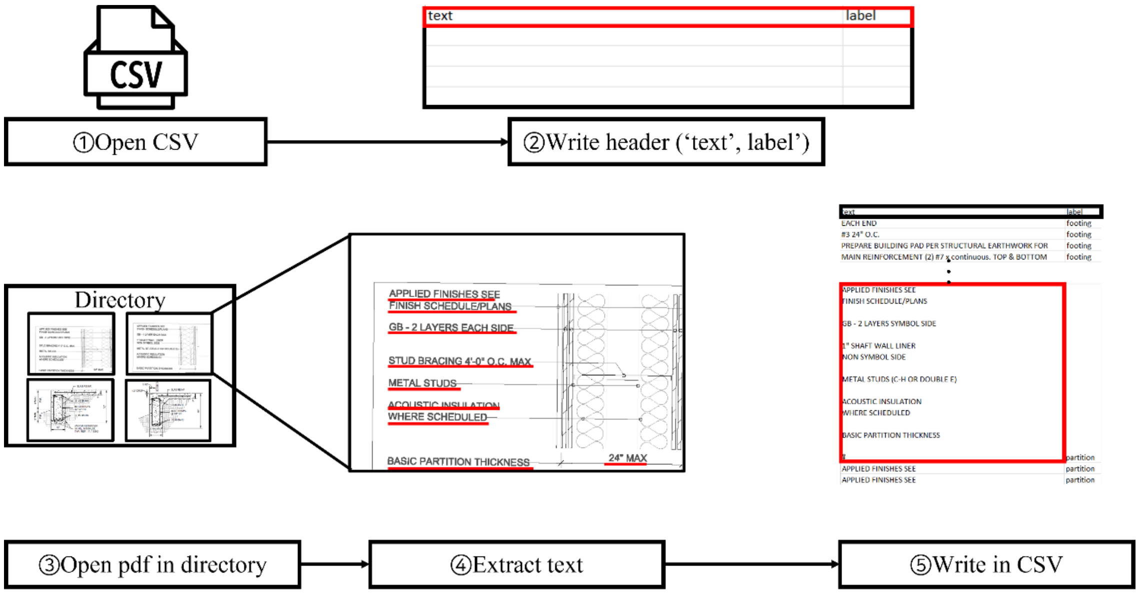


Figure 18 Structure of Text Extraction Model

The overall workflow of the model is shown in Figure 19. In the beginning, it creates a new CSV file in writing mode and opens it with an imported CSV module. Headers of the CSV cells are written after creating the file. With this process, columns named 'text' and 'label' are created and written into the file as header. After creating a CSV file with a header, the model opens all single component detail sections drawing a pdf file located into the directory. The model then scans each file and extracts the text with the pdfminer module. Some files cannot be extracted because texts are handwritten or saved in image format, not text format. In these cases, texts are handwritten and manually typed into a CSV file. After extraction, each text's label names are manually provided according to the RSMeans system components description. These texts and label data are then written in the CSV file. The output of this model is a CSV file in Figure 12 format. The python code for this model is given in Appendix A-2.



text	label
EACH END	footing
#3 24" O.C.	footing
PREPARE BUILDING PAD PER STRUCTURAL EARTHWORK FOR	footing
MAIN REINFORCEMENT (2) #7 x continuous. TOP & BOTTOM	footing
APPLIED FINISHES SEE FINISH SCHEDULE/PLANS GB - 2 LAYERS SYMBOL SIDE 1" SHAFT WALL LINER NON SYMBOL SIDE METAL STUDS (C-H OR DOUBLE E) ACOUSTIC INSULATION WHERE SCHEDULED BASIC PARTITION THICKNESS #	partition
APPLIED FINISHES SEE	partition
APPLIED FINISHES SEE	partition

Figure 19 Workflow of Text Extraction Model

Text Classification Model

This section presents the architecture and workflow of the text classification model. The overall work procedure of those two modules is shown in Figure 20. Text preprocessing module and LSTM based neural network are two major elements comprising this model. The former works as a text preprocessor which refine the text and label data into a numerical form which enables the neural network to process the data. Meanwhile, the latter is an inferential base, where the classification of the processed data and making a decision occurs. The detailed structure and workflow of the model are presented in the following subchapter. In addition, the python-based codes are presented in Appendix A-3.

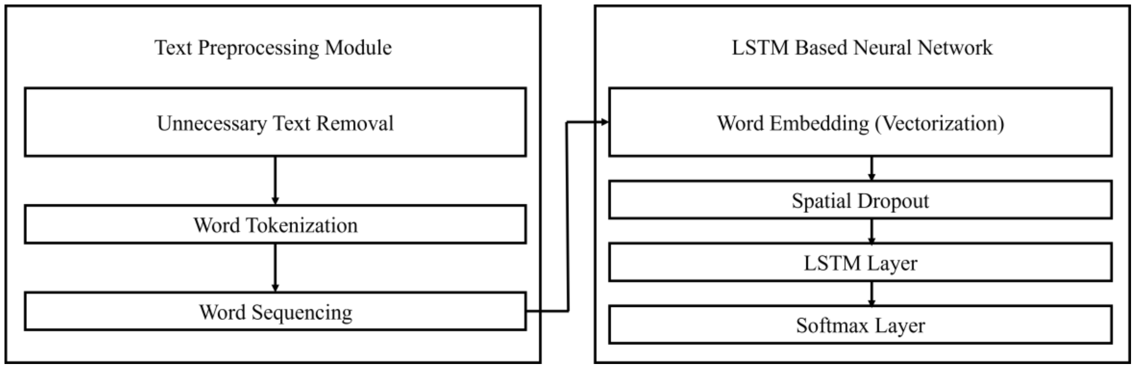


Figure 20 Workflow of the Text Classification Model

Text Preprocessing Module

The text preprocessing module transforms the text data into the format suitable for the LSTM based module to process the training and testing. The overall structure and data pipeline are presented in Figure 21.

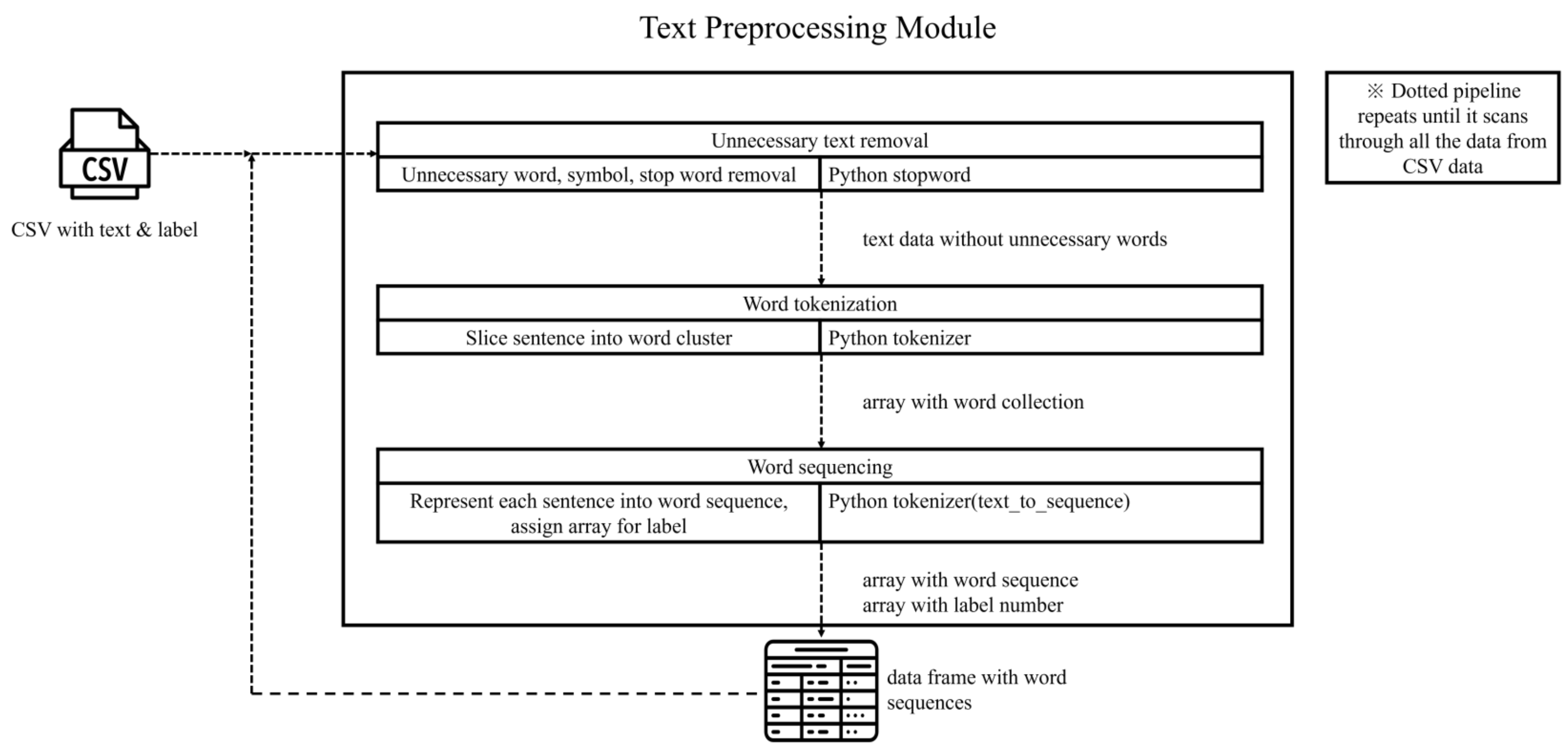
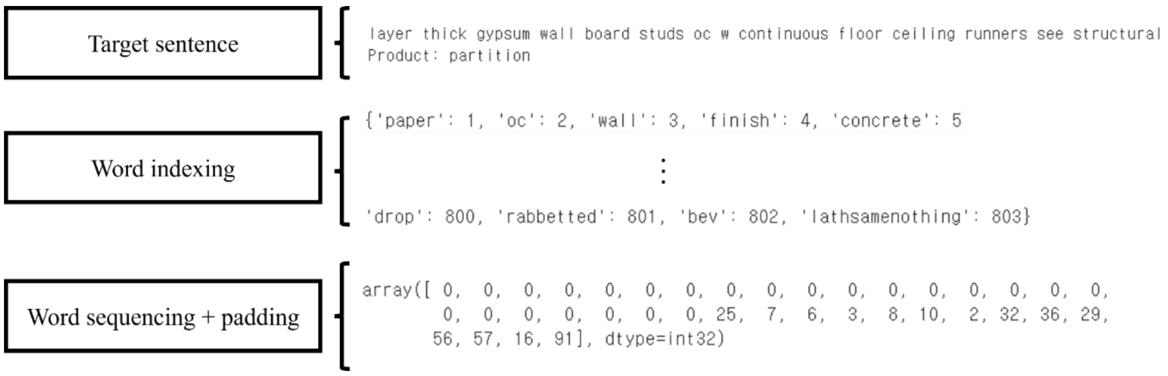

Text Preprocessing Module
CSV
CSV with text & label
Unnecessary text removal
Unnecessary word, symbol, stop word removal
Python stopword
text data without unnecessary words
Word tokenization
Slice sentence into word cluster
Python tokenizer
array with word collection
Word sequencing
Represent each sentence into word sequence, assign array for label
Python tokenizer(text_to_sequence)
array with word sequence
array with label number
data frame with word sequences
※ Dotted pipeline repeats until it scans through all the data from CSV data

Figure 21 Structure and Data Pipeline of Text Preprocessing Module

The first element of the module is the unnecessary text removal cell. This element removes words that rarely bear the information needed to train and test the model. In this project, punctuation marks and special symbols such as brackets, numbers, single letters, and others are unnecessary words. In addition, stopwords are also deleted with this element. Stopwords are commonly used word components that rarely contain helpful information. Examples of stopwords are article words such as 'a', 'the,' or auxiliary verbs such as 'am,' 'are,' or 'is.' In this project, the Stopword library is imported into the model and compared with the text data to remove them.

The unnecessary text removal cell is followed by the word tokenizer. Word tokenizer slices the sentence into word clusters called tokens and stores them into word arrays. Space, comma, period mark, asterisk, and other marks work as benchmarks for tokenizing words in this project. Python-based tokenizer modules are utilized to construct word tokenization cells.

Word sequencing cell is the subsequent element of the module. Word sequencing refers to assigning a unique number to each word token (word indexing) and representing text data into a number array (word sequencing.) After sequencing, 0 are padded to synchronize the array length among entire word arrays because LSTM based module perceives a single typed length of the array when training and testing. An example of word sequencing is presented in Figure 22. In the target sentence, each word has a unique index: 'layer' – 25, 'thick' – 7, 'gypsum' – 6, and so on. The index numbers are located into the number array and padded with 0s. The text_to_sequence() function from the tokenizer module is the primary function for constructing a word sequencing cell. Meanwhile, the label data also transforms into a numerical array because LSTM based module recognizes only numerical array type, not word data. This transformation is done by the get_dummies() function from the pandas library.



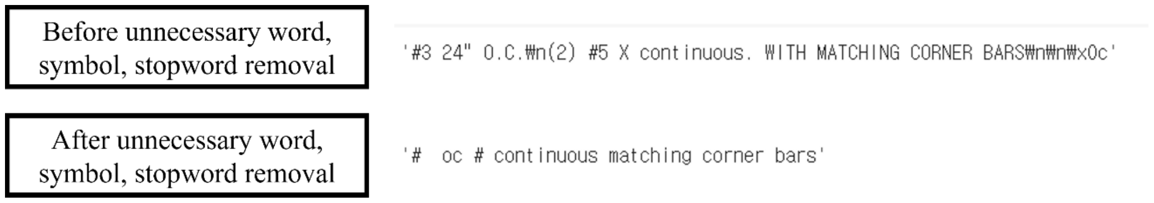
```
layer thick gypsum wall board studs oc w continuous floor ceiling runners see structural
Product: partition
```

```
{'paper': 1, 'oc': 2, 'wall': 3, 'finish': 4, 'concrete': 5
                              ⋮
'drop': 800, 'rabbetted': 801, 'bev': 802, 'lathsamenothing': 803}
```

```
array([ 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0, 25,  7,  6,  3,  8, 10,  2, 32, 36, 29,
       56, 57, 16, 91], dtype=int32)
```

Figure 22 Example of Word Sequencing Process

The data from text extraction undergoes the text preprocessing module in sequential order. This begins with removing unnecessary words, symbols, and stopwords are eliminated. In addition, large size spacings are altered into a single space for future generations processing. Examples of the word symbol removal process are shown in Figure 23. After removing, the word tokenizer divides a sentence into multiple word arrays. As a result of this process, 803 unique tokens are produced. Sets of word arrays from the tokenization process transform into word sequences. Afterward, the label data are assigned a unique number array. In this project, array distributions are composed with 1 and 0 forms, where 1 represents the true for each label while 0 represents false for the label. Example forms for the array are as follows: Standard foundation – [0,1,0], exterior wall – [1,0,0], partition – [0,0,1]. These word sequence and label arrays are stacked into a data frame. The above procedure is repeated until it covers all the data from CSV data.


Before unnecessary word, symbol, stopword removal
After unnecessary word, symbol, stopword removal

```
'#3 24" O.C.₩n(2) #5 X continuous. WITH MATCHING CORNER BARS₩n₩n₩x0c'
```

```
'#  oc # continuous matching corner bars'
```

Figure 23 Example of Unnecessary Word, Symbol, Stopword Removal Process

LSTM Based Neural Network

LSTM based module is a core part of the text classification model where training and validation for decision-making occur. The module is a sequential model which consists of several types of layers in a set of order: embedding layers, spatial dropout layers, LSTM layers, and dense layers. The sequential model and each layer are built with the TensorFlow Keras library. This library integrates multiple cells into layers to construct a building block for the neural network. A detailed structure of LSTM based module is shown in Figure 24.

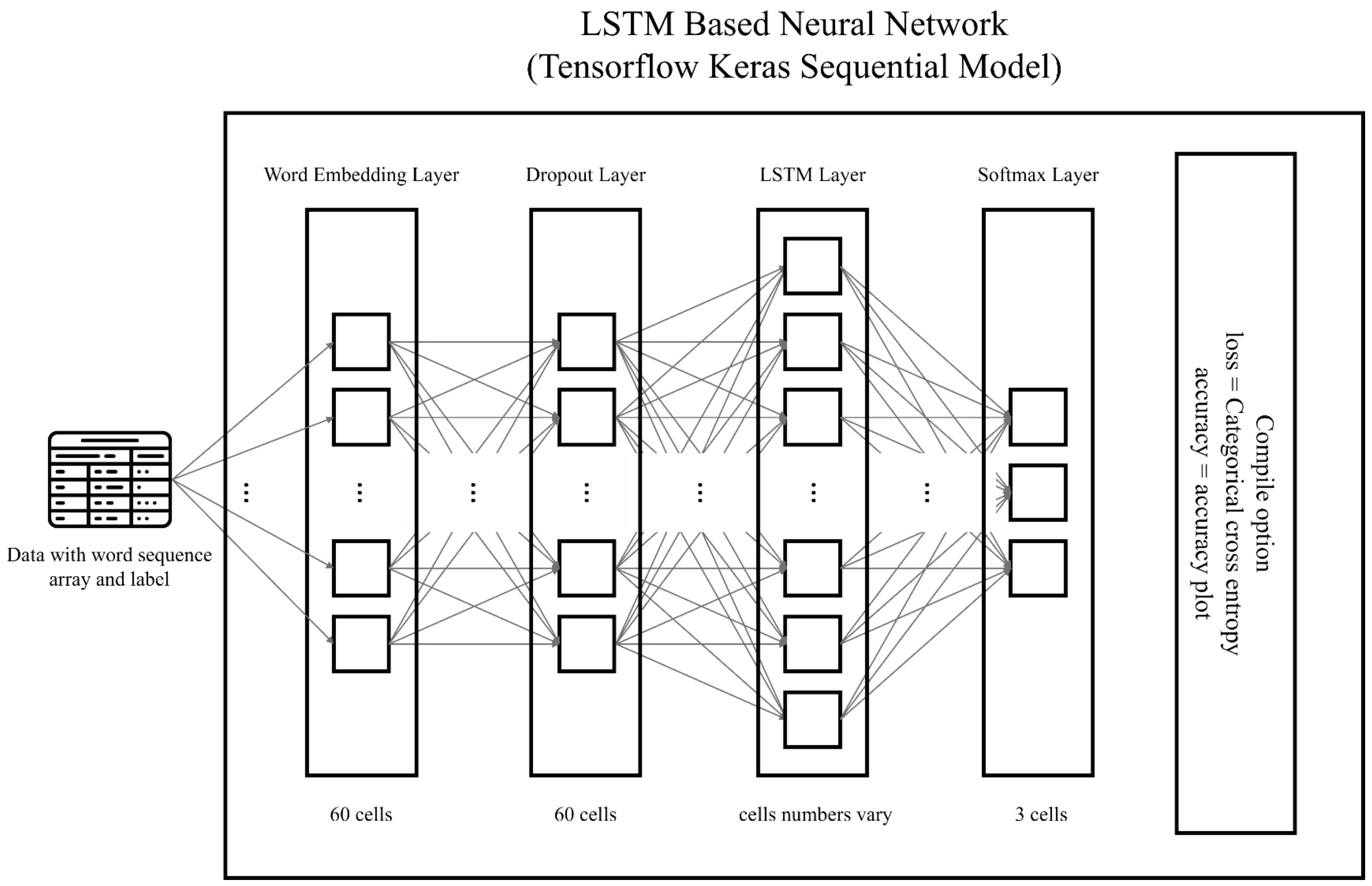


Figure 24 Detailed Structure of LSTM Based Module

In the embedding layer, word sequencing arrays are mapped with numerical vector values and stored in individual cells called the embedding cells. The distance between vectors represents how close the meaning of each word sequence is. In other words, if the number of embedding dimensions increases, the parameter to represent the closeness of the word becomes more sensitive. In this project, the size of this layer equals the multiplication of word sequencing array and word embedding dimension, where maximum word sequencing length is 38, and word embedding length is 60.

The next layer of the module is the dropout layer. The cell in the layer selectively deactivates input layers to LSTM during the training process. This layer prevents the model from being overfitted. An overfitting problem happens when the model is so flexible that it even follows the undesirable error value for training (James et al., 2021). This phenomenon might cause the model to perform poorly. In this project, the dropout rate is set up as 0.2, meaning 20% of layers are excluded in each training process.

After the dropout layer, LSTM layers are followed. Each LSTM cell in the layer utilizes previous memory while training the model. In other words, this LSTM model decides whether the previous data should be retained or be discarded for training. In this project, cell numbers of the LSTM network will be changed, and the evaluation scores from each change will be compared for the best model performance.

The dense layer is located after the LSTM layer for classification. Individual cells in this layer calculate the statistical probability of the target data being the labeled building component. In this project, as the classification label is more than 2, non-binary classification, the softmax layer was adopted (Bridle, 1990). This layer is designed to produce the probability distribution of given labels eventually. In other words, when the input data is given, each probability represents the possibility that the data belonging to a specific label type.

Above layers are added to the sequential model and compiled. In this process, the output loss function type is set up. As this project aims to plot the loss of multi-label classification problems, the categorical cross-entropy function is selected. The categorical cross-entropy function quantifies the difference between prediction and target data probability distribution.

The neural network procedure begins with the word sequence and label data from text preprocessing module output. This data works as an input for the neural network. These input data pass through each cell to subsequent cells in the next layer. Individual cells decide to pass or abandon the data according to their cell characteristics. This procedure eventually reaches the softmax layer, classifying the given data according to the building component label.

CHAPTER IV

MODEL EVALUATION

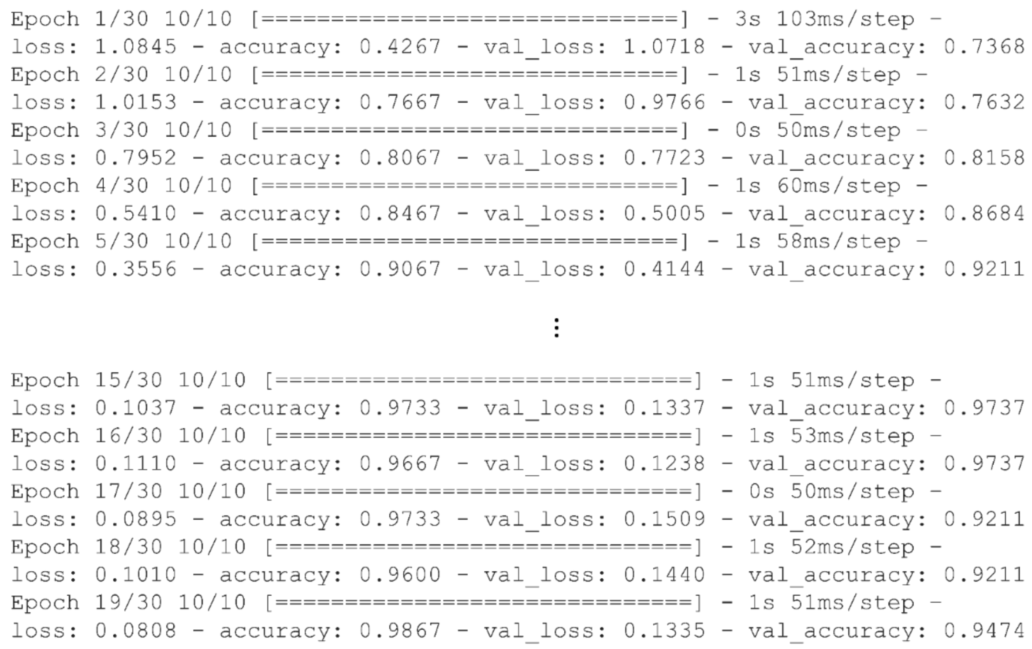
This section presents the model performance result and analysis of the outcomes.

Model Training and Testing

This section describes how models are trained and tested. Moreover, the output of each process will be introduced.

Once the training and testing data pipeline is created, the epoch and batch size are assigned to train the model. The number of epochs refers to the number of training models, while batch size means the unit number of samples taken from an entire training set. In addition, validation split rates are given according to the validation set method. This rate refers to the rate of assigning validation set from training data. In This research, 20% of training data are set apart as a validation set.

As a result of this process, it plots training/ validation loss and training/validation accuracy in each epoch. The sample progress when the number of LSTM layers is 80 is shown in Figure 25. This study is designed to stop the process once the validation loss value does not show significant changes. The model tolerates until the validation loss within three epochs is the same for 10^{-4} digits. The sample graph of the loss and accuracy is shown in Figure 26 and Figure 27. In each graph, the x-axis refers to the number of epochs, while the y-axis represents the percentage of loss and accuracy in each epoch.



```
Epoch 1/30 10/10 [==============================] - 3s 103ms/step -
loss: 1.0845 - accuracy: 0.4267 - val_loss: 1.0718 - val_accuracy: 0.7368
Epoch 2/30 10/10 [==============================] - 1s 51ms/step -
loss: 1.0153 - accuracy: 0.7667 - val_loss: 0.9766 - val_accuracy: 0.7632
Epoch 3/30 10/10 [==============================] - 0s 50ms/step -
loss: 0.7952 - accuracy: 0.8067 - val_loss: 0.7723 - val_accuracy: 0.8158
Epoch 4/30 10/10 [==============================] - 1s 60ms/step -
loss: 0.5410 - accuracy: 0.8467 - val_loss: 0.5005 - val_accuracy: 0.8684
Epoch 5/30 10/10 [==============================] - 1s 58ms/step -
loss: 0.3556 - accuracy: 0.9067 - val_loss: 0.4144 - val_accuracy: 0.9211

                                  ⋮

Epoch 15/30 10/10 [==============================] - 1s 51ms/step -
loss: 0.1037 - accuracy: 0.9733 - val_loss: 0.1337 - val_accuracy: 0.9737
Epoch 16/30 10/10 [==============================] - 1s 53ms/step -
loss: 0.1110 - accuracy: 0.9667 - val_loss: 0.1238 - val_accuracy: 0.9737
Epoch 17/30 10/10 [==============================] - 0s 50ms/step -
loss: 0.0895 - accuracy: 0.9733 - val_loss: 0.1509 - val_accuracy: 0.9211
Epoch 18/30 10/10 [==============================] - 1s 52ms/step -
loss: 0.1010 - accuracy: 0.9600 - val_loss: 0.1440 - val_accuracy: 0.9211
Epoch 19/30 10/10 [==============================] - 1s 51ms/step -
loss: 0.0808 - accuracy: 0.9867 - val_loss: 0.1335 - val_accuracy: 0.9474
```

Figure 25 Sample Training/ Testing Loss and Accuracy in Each Epoch (LSTM_LAYER_NUM = 80)

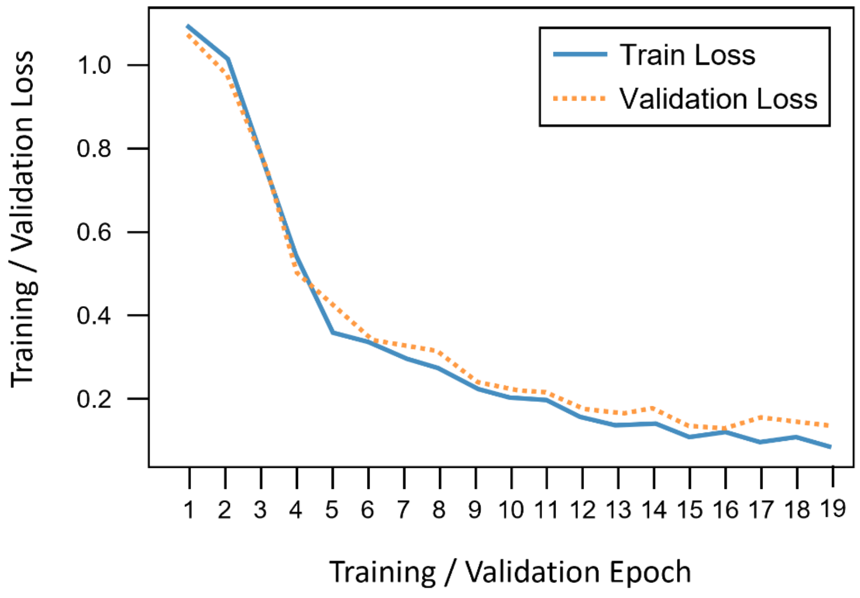


Figure 26 Sequential Change of Loss in Training and Testing Sets (LSTM Layer Number=80)

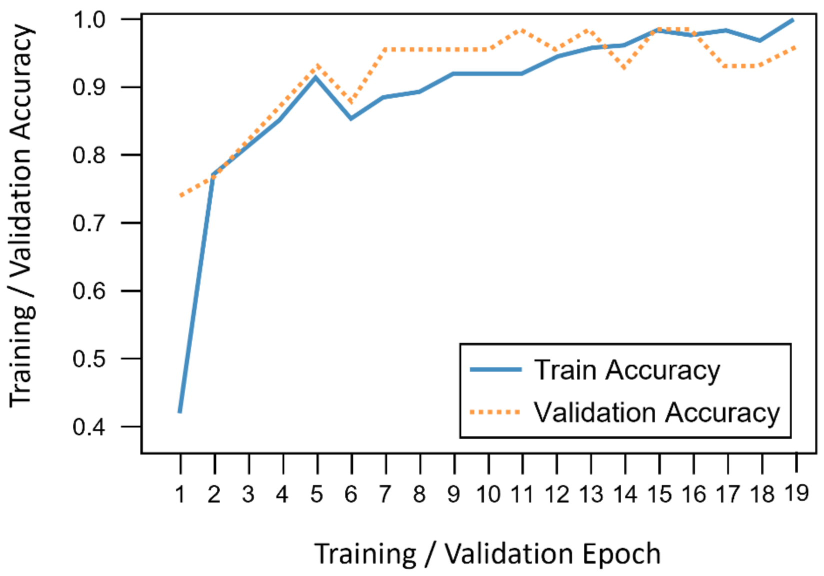


Figure 27 Sequential Change of Accuracy in Training and Testing Sets (LSTM Layer Number=80)

After training and validation, spared testing data is used to monitor how accurate the model predicts when the new dataset is given. In this project, 40% of the entire data set is spared for testing the model. This testing data undergoes the testing pipeline of the prototype and produces WP decision results or prediction results.

To evaluate the accuracy of the classification, the prediction results from the testing dataset and the actual label value of the data are compared. Before the comparison, prediction values are manipulated into 1 and 0 array forms. This is because the label data are in 1 and 0 array representation, while prediction values are shown in an array of the prediction probability distribution form. The format is synchronized by transforming the largest probability value among the array into one and the rest of the values into 0. Comparison becomes possible after this transformation. Once similarities are compared, the evaluation matrix values (accuracy, precision, recall, and f-1 scores) are calculated using the sklearn metrics library for python language. Table 5 illustrates loss, accuracy, evaluation matrix values when each parameter is changed. Especially, the number of LSTM cells, or LSTM_LAYER_NUM, is changed to find out the best model that best fits the dataset.

Table 5 Training/ Test Result of the in Different LSTM cell Number

LSTM_CELL_NUM	50	60	70	80	90
LOSS	0.821627	0.432442	0.210753	0.229157	0.382901
ACCURACY	0.896825	0.928571	0.936508	0.952381	0.928571
PRECISION	0.907619	0.932181	0.936822	0.952328	0.930525
RECALL	0.901163	0.931466	0.939042	0.953398	0.931466
F-1	0.896013	0.928468	0.93649	0.952231	0.928281

Table 6 Evaluation Matrix Values (LSTM Cell Number=80)

Evaluation Matrix	Value
Loss	0.229157
Accuracy	0.952381
Precision	0.952328
Recall	0.953398
F-1 Score	0.952231

Analysis of the Result

This section presents the output analysis from the training and testing procedure of the model.

As shown in Figure 25 to Figure 27, it was found that the training/ testing loss gradually decreases, whereas training/testing accuracy increases over time. This result demonstrates the characteristics of DL, especially the LSTM model, which acquires information from the previous attempt and sequentially improves the model. Although several fluctuation points are spotted, the sequential improvement of the model shows that this model can accept test data for the WP decision.

The LSTM cell numbers were calibrated in this testing result to examine the best model for the given data. The LSTM cell number and the subsequent evaluation matrix values are shown in Table 5. It is discovered that loss value keeps decreasing while accuracy increases as the number of cell increase. The loss becomes the lowest when the LSTM cell number becomes 70, while accuracy becomes the highest among trials when the LSTM cell number reaches 80. This indicates that the more complex the LSTM neural network is, the better the prediction performance tends to be. However, the large number of LSTM cells does not guarantee higher accuracy. As the number of LSTM cells goes beyond 80, the accuracy value does not show a significant increase, whereas the loss becomes more prominent when the cell number exceeds 90. We could assume that this phenomenon occurred because of the overfitting problem, meaning that models with too complicated neural networks fit into undesired data. As a result, the best model for the automated WP decision model with given data is the LSTM cell number with 80.

Table 6 presents the evaluation matrix value when LSTM cell number was given as 80. Comparing the previous work of Che-Wen et al. (2021), which is presented Table 2, the prototype produces significantly accurate decision. This is because the accuracy, precision, recall value are close to 95%, and F-1 score in the proposed framework is 1% larger than the previous research. Moreover, as the previous model involves bigger number of data (35821), this study has potential to be more precise model once the dataset amount increases.

CHAPTER V

CONCLUSION

This research aimed at augmenting the abilities of inexperienced construction teams working on WP composition for efficient construction work project planning. Such a procedure entailed several problems: WBS composition difficulties, inconsistency of format and convention of WP, and time and accuracy problems when dealing with large and complex projects. As a solution, this study proposed the development of a prototype that automates WP's decision-making process from detail section construction drawing within consistent criteria. In order to accomplish the research goal, three research objectives are established: To organize data within RSMeans criteria (Objective 1), building up the prototype structure (Objective 2), and evaluate the decision result (Objective 3). The first objective was achieved by collecting detail section drawings (Plan 1) and analyzing their features (Plan 2). In addition, the prototype architecture was successfully built by building up text extraction model (Plan 3) and text classification model (Plan 4). Moreover, the test result was evaluated by proceeding training and testing for the prototype (Plan 5) and analyzing test result of approximately 95.24% accuracy and 22.91% loss (Plan 6.)

The benefit of this study was that it can potentially assist the manual process of collecting text data from the set of detail section drawings and deciding which WP format and category to use. In addition, consistent WP composition was expected when different participants use the prototype. It will also significantly save time and reduce error compared to human performance. This attempt would be beneficial because this can be integrated into any construction project activities which involve WBS and WP organizations. Moreover, if the target text data other than construction drawing, such as medical records, this model would also be applicable for constructing a decision-making model for other fields of study.

Future studies can provide an improved model by utilizing images, symbols, and text in construction drawings containing multiple building components. In addition, prospective research can be extended to the full version of RSMeans Assemblies Costs data as WP decision criteria and making the decision for more specific assembly division, and various types of projects (i.e., residential project, heavy civil project.)

REFERENCES

Bosch-Rekveldt, M., Jongkind, Y., Mooi, H., Bakker, H., & Verbraeck, A. (2011). Grasping project complexity in large engineering projects: The TOE (Technical, Organizational and Environmental) framework. *International Journal of Project Management*, *29*(6), 728-739. https://doi.org/https://doi.org/10.1016/j.ijproman.2010.07.008

Bridle, J. (1990). Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition. In (pp. 227-236). https://doi.org/10.1007/978-3-642-76153-9_28

Brown, W. C., & Dorfmueller, D. P. (2019). Print reading for construction : residential and commercial : write-in text with 140 large prints. by Walter C. Brown, Daniel P. Dorfmueller. 7th ed (7th ed.) [Non-fiction]. *Goodheart-Willcox Company, Inc.* https://proxy.library.tamu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat03318a&AN=tamug.4546729&site=eds-live

Bughin, J., Hazan, E., Ramaswamy, S., Chui, M., Allas, T., Dahlstrom, P., Henke, N., & Trench, M. (2017). Artificial Intelligence The Next Digital Frontier? In (pp. 71): *Mckinsey & Company*.

Calabrese, A., Camaioni, M., & Piervincenzi, G. (2019). Advanced Work Packaging in capital projects: a standardized model for EPC Contractors [Article]. *Journal of Modern Project Management*, *7*(3), 1-21. https://doi.org/10.19255/JMPM02102

Ce, P., & Tie, B. (2020). An Analysis Method for Interpretability of CNN Text Classification Model. *Future Internet*, *12*(12). https://doi.org/10.3390/fi12120228

Charette, R., & Marshall, H. (1999). UNIFORMAT II elemental classification for building specifications, cost estimating, and cost analysis. Gaithersburg, MD : *U.S. Dept. of Commerce, National Institute of Standards and Technology*, 1999. Retrieved from https://purl.fdlp.gov/GPO/gpo98355

Che-Wen, C., Shih-Pang, T., & Jhing-Fa, W. (2021). Outpatient Text Classification System Using LSTM [Article]. *Journal of Information Science & Engineering*, *37*(2), 365-379. https://doi.org/10.6688/JISE.202103_37(2).0006

Gordian. (2017). Assemblies costs with RSMeans data (44th ed.) [Statistics]. *Gordian RSMeans Data.* https://proxy.library.tamu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat03318a&AN=tamug.5393477&site=eds-live

Graves, A. (2012). Supervised Sequence Labelling with Recurrent Neural Networks (Vol. 385). https://doi.org/10.1007/978-3-642-24797-2

Hines, V. (2021). The State of Project Management 2021 Annual Report. *Wellington*.

Hu, Y., & Castro-Lacouture, D. (2019). Clash Relevance Prediction Based on Machine Learning. *Journal of Computing in Civil Engineering*, *33*(2nd), 04018060. https://doi.org/10.1061/(ASCE)CP.1943-5487.0000810

Huang, W., & Zheng, H. (2018, 10/18/2018-10/20/2018). Architectural Drawings Recognition and Generation through Machine Learning. *Recalibration : on imprecision and infidelity*, Universidad Iberoamericanan, Mexico City.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). An Introduction to Statistical Learning, with Applications in R. https://doi.org/10.1007/978-1-0716-1418-1

Jones, C. (2007). Creating an effective WBS with facilitated team involvement *PMI Global Congress 2007* - North America, Atlanta, Georgia.

Kelleher, J. D. (2019). *Deep learning* [Bibliographies Online Non-fiction Electronic document]. *The MIT Press*. https://proxy.library.tamu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat08996a&AN=tamu.8839f2be.e315.3078.bff0.c4043ea739d0&site=eds-live
http://proxy.library.tamu.edu/login?url=https://ebookcentral.proquest.com/lib/tamucs/detail.action?docID=5855529

Khan, I. U., Afzal, S., & Lee, J. W. (2022). Human Activity Recognition via Hybrid Deep Learning Based Model. *Sensors (Basel, Switzerland)*, *22*(1). https://doi.org/10.3390/s22010323

Kim, S., & Gibson, G. (2002). ASSESSMENT OF CII KNOWLEDGE IMPLEMENTATION AT THE ORGANIZATIONAL LEVEL.

Krstinic, D., Braović, M., Šerić, L., & Božić-Štulić, D. (2020). Multi-label Classifier Performance Evaluation with Confusion Matrix. https://doi.org/10.5121/csit.2020.100801

Kumar, K., Zindani, D., & Davim, J. P. (2019). Intelligent Manufacturing. In K. Kumar, D. Zindani, & J. P. Davim (Eds.), *Industry 4.0: Developments towards the Fourth Industrial Revolution* (1st ed., pp. 71). Springer Singapore. https://doi.org/https://doi.org/10.1007/978-981-13-8165-2_1

Merrow, E. W. (2011). Industrial megaprojects: concepts, strategies, and practices for success. *John Wiley & Sons*.

Mewada, H. K., Patel, A. V., Chaudhari, J., Mahant, K., & Vala, A. (2020). Automatic room information retrieval and classification from floor plan using linear regression model [Original Paper]. *International Journal on Document Analysis and Recognition (IJDAR)*, *23*(4th), 253. https://doi.org/https://doi.org/10.1007/s10032-020-00357-x

Nof, S. Y. (2009). Automation: What It Means to Us Around the World. In S. Y. Nof (Ed.), *Springer Handbook of Automation* (1st ed., pp. 13-52). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-78831-7_3

Pan, Y., & Zhang, L. (2021). Roles of artificial intelligence in construction engineering and management: A critical review and future trends [Review Article]. *Automation in Construction*, *122*, 103517. https://doi.org/10.1016/j.autcon.2020.103517

PMI. (2017). A guide to the project management body of knowledge (Sixth edition. ed.) [Bibliographies Non-fiction]. *Project Management Institute*, Inc. https://proxy.library.tamu.edu/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=cat08996a&AN=tamu.98b1a6c3.495e.3925.bc03.d3817eba0baf&site=eds-live

Rho, J., Lee, H. S., & Park, M. (2020). Automated BIM generation using drawing recognition and line-text extraction. *Journal of Asian Architecture and Building Engineering*, *20*(6), 747. https://doi.org/10.1080/13467581.2020.1806071

Simpson, F. A. (2009). Cibolo Fire Station NO.2 In. *AIA* ,144 Landa Street, Suite 153 New Braunfels, Texas: Archimedia PLLC.

Williams, T. (2005). Assessing and moving on from the dominant project management discourse in the light of project overruns. *IEEE Transactions on engineering management*, *52*(4), 497-508.

Wu, X., Zhao, Y., Radev, D., & Malhotra, A. (2020). Identification of patients with carotid stenosis using natural language processing. *European Radiology*, *30*(7), 4125-4133. https://doi.org/10.1007/s00330-020-06721-z

Zhou, C., Sun, C., Liu, Z., & Lau, F. C. M. (2015). A C-LSTM Neural Network for Text Classification. In. arXiv.

Zhou, Y. (2022). Natural Language Processing with Improved Deep Learning Neural Networks [Article]. *Scientific Programming*, 1-8. https://doi.org/10.1155/2022/6028693

APPENDIX A

APPENDIX A CODE STRUCTURE AND RESULT

A-1 Word Frequency Plot

```
from google.colab import drive
drive.mount('/content/drive')

!pwd
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/My Drive/Colab Notebooks/Building Component Identification
/content/drive/MyDrive/Colab Notebooks/Building Component Identification/individual
/content/drive/MyDrive/Colab Notebooks/Building Component Identification/individual
'separate_exterior wall.csv'   separate_footing.csv   separate_partition.csv
```

```
import pandas as pd
df = pd.read_csv('separate_partition.csv')
df.info()

%cd /content/drive/MyDrive/Colab\ Notebooks/Building\ Component\
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103 entries, 0 to 102
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   text    103 non-null    object
 1   label   103 non-null    object
dtypes: object(2)
memory usage: 1.7+ KB
```

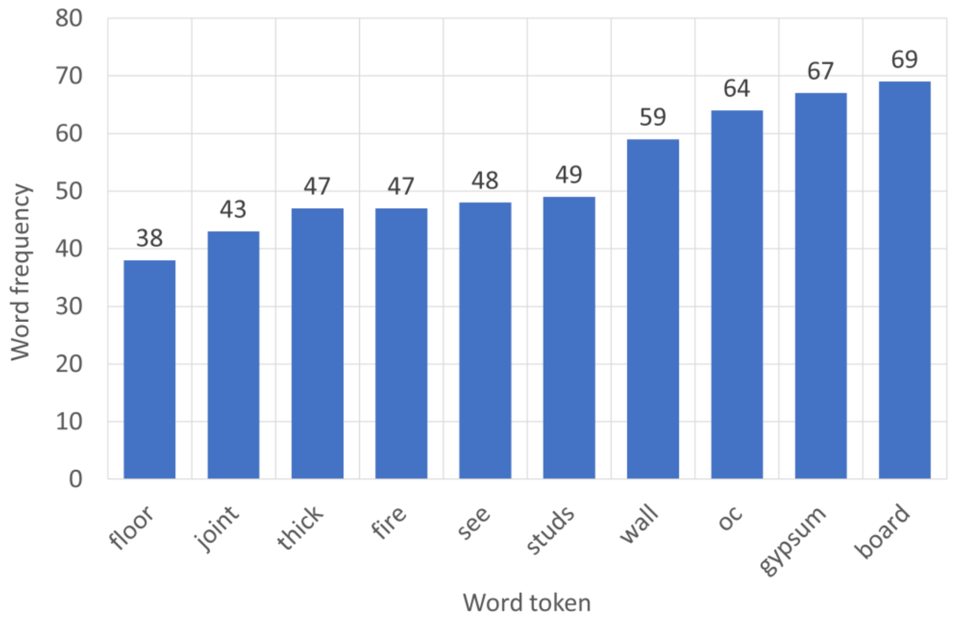
```
import re
import nltk
nltk.download('stopwords')
STOPWORDS = set(stopwords.words('english'))
from nltk.corpus import stopwords
df = df.reset_index(drop=True)
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))

def clean_text(text):
  text= text.lower()
  text = REPLACE_BY_SPACE_RE.sub('',text)
  text = BAD_SYMBOLS_RE.sub('', text)
  text = text.replace('x', '')
  text = ' '.join(word for word in text.split() if word not in ST
OPWORDS)
  return text
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
from nltk import word_tokenize
from keras.preprocessing.text import Tokenizer
tokenizer = Tokenizer(num_words=122, filters='!"#$%&()*+,-./:;<=>
?@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(df['text'].values)
word_index = tokenizer.word_index

print('Found %s unique tokens.' % len(word_index))
print("\n")
print(word_index)
```

```
Found 288 unique tokens.
```

```
from collections import OrderedDict
import matplotlib.pyplot as plt
top10 = OrderedDict(sorted(tokenizer.word_counts.items(),key=lamb
da t : t[1])[-10:])

bar=plt.bar(top10.keys(),top10.values())
for rect in bar:
    height = rect.get_height()
    plt.text(rect.get_x() + rect.get_width()/2.0, height, '%.1f'
% height, ha='center', va='bottom', size = 12)
```



A-2 Text Extraction Model

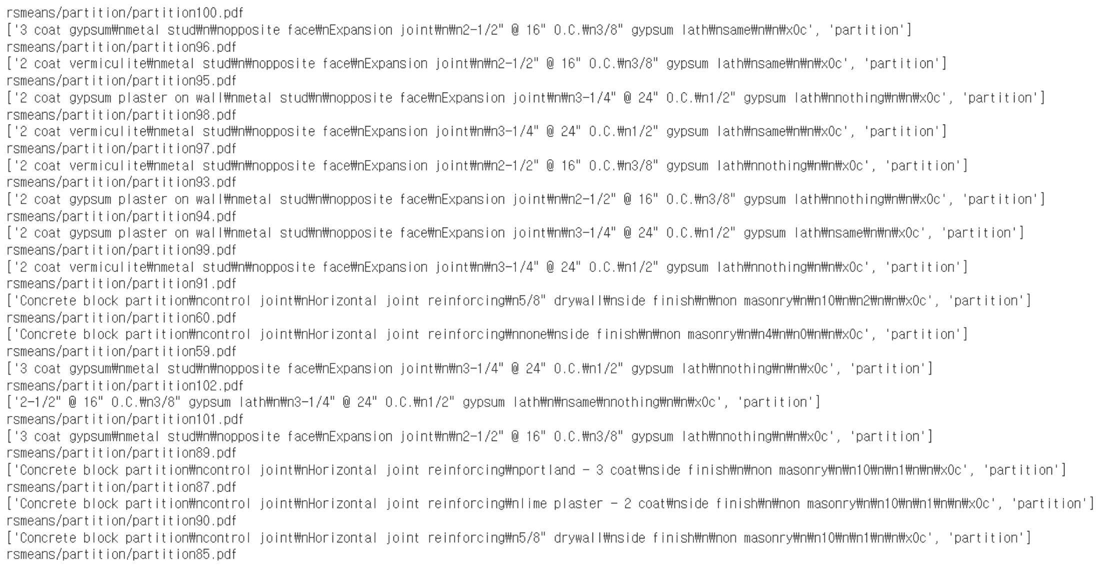
```
from google.colab import drive
drive.mount('/content/drive')


%cd /content/drive/MyDrive/Colab\ Notebooks/Building\ Component\
Identification
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/MyDrive/Colab Notebooks/Building Component Identification
```

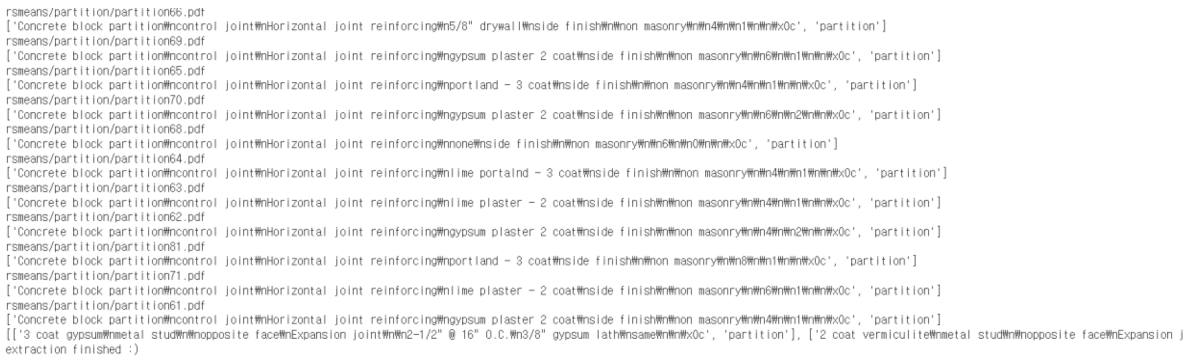
```
! pip install pdfminer.six

#import modules
import pdfminer
from pdfminer import high_level
import os
import csv
```

```
Requirement already satisfied: pdfminer.six in /usr/local/lib/python3.7/dist-packages (20211012)
Requirement already satisfied: cryptography in /usr/local/lib/python3.7/dist-packages (from pdfminer.six) (36.0.1)
Requirement already satisfied: chardet in /usr/local/lib/python3.7/dist-packages (from pdfminer.six) (3.0.4)
Requirement already satisfied: cffi>=1.12 in /usr/local/lib/python3.7/dist-packages (from cryptography->pdfminer.six) (1.15.0)
Requirement already satisfied: pycparser in /usr/local/lib/python3.7/dist-packages (from cffi>=1.12->cryptography->pdfminer.six) (2.21)
```

```
with open('drawing text_integrated.csv','w', encoding='UTF8', new
line='') as f:
  writer = csv.writer(f)
  header = ['text','label']
  #write header
  writer.writerow(header)


data = []
label_name="partition"
file_list = os.listdir('rsmeans/'+label_name)

for i in file_list:
  #extract word with pdfminer.high_level
  path = 'rsmeans/'+label_name + '/'+i
  extracted_text = high_level.extract_text(path,"",[0])
  print('%s' % (path))
  extracted_text_mod = extracted_text.strip()
  list = [extracted_text,label_name]
  print(list)
  data.append(list)

print(data)

with open('drawing text_integrated.csv','a', encoding='UTF8', new
line='') as f:
  writer = csv.writer(f)

  #write multiple rows
  writer.writerows(data)

print("extraction finished :)")
```



```
rsmeans/partition/partition100.pdf
['3 coat gypsum₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩nsame₩n₩n₩x0c', 'partition']
rsmeans/partition/partition96.pdf
['2 coat vermiculite₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩nsame₩n₩n₩x0c', 'partition']
rsmeans/partition/partition95.pdf
['2 coat gypsum plaster on wall₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n3-1/4" @ 24" O.C.₩n1/2" gypsum lath₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition98.pdf
['2 coat vermiculite₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n3-1/4" @ 24" O.C.₩n1/2" gypsum lath₩nsame₩n₩n₩x0c', 'partition']
rsmeans/partition/partition97.pdf
['2 coat vermiculite₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition93.pdf
['2 coat gypsum plaster on wall₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition94.pdf
['2 coat gypsum plaster on wall₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n3-1/4" @ 24" O.C.₩n1/2" gypsum lath₩nsame₩n₩n₩x0c', 'partition']
rsmeans/partition/partition99.pdf
['2 coat vermiculite₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n3-1/4" @ 24" O.C.₩n1/2" gypsum lath₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition91.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩n5/8" drywall₩nside finish₩n₩non masonry₩n₩n10₩n₩n2₩n₩n₩x0c', 'partition']
rsmeans/partition/partition60.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nnone₩nside finish₩n₩non masonry₩n₩n4₩n₩n0₩n₩n₩x0c', 'partition']
rsmeans/partition/partition59.pdf
['3 coat gypsum₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n3-1/4" @ 24" O.C.₩n1/2" gypsum lath₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition102.pdf
['2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩n₩n3-1/4" @ 24" O.C.₩n1/2" gypsum lath₩n₩nsame₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition101.pdf
['3 coat gypsum₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩nnothing₩n₩n₩x0c', 'partition']
rsmeans/partition/partition89.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nportland - 3 coat₩nside finish₩n₩non masonry₩n₩n10₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition87.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nlime plaster - 2 coat₩nside finish₩n₩non masonry₩n₩n10₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition90.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩n5/8" drywall₩nside finish₩n₩non masonry₩n₩n10₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition85.pdf
```



```
rsmeans/partition/partition66.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩n5/8" drywall₩nside finish₩n₩non masonry₩n₩n4₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition69.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩ngypsum plaster 2 coat₩nside finish₩n₩non masonry₩n₩n6₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition65.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nportland - 3 coat₩nside finish₩n₩non masonry₩n₩n4₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition70.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩ngypsum plaster 2 coat₩nside finish₩n₩non masonry₩n₩n6₩n₩n2₩n₩n₩x0c', 'partition']
rsmeans/partition/partition68.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nnone₩nside finish₩n₩non masonry₩n₩n6₩n₩n0₩n₩n₩x0c', 'partition']
rsmeans/partition/partition64.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nlime portalnd - 3 coat₩nside finish₩n₩non masonry₩n₩n4₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition63.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nlime plaster - 2 coat₩nside finish₩n₩non masonry₩n₩n4₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition62.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩ngypsum plaster 2 coat₩nside finish₩n₩non masonry₩n₩n4₩n₩n2₩n₩n₩x0c', 'partition']
rsmeans/partition/partition81.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nportland - 3 coat₩nside finish₩n₩non masonry₩n₩n8₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition71.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩nlime plaster - 2 coat₩nside finish₩n₩non masonry₩n₩n6₩n₩n1₩n₩n₩x0c', 'partition']
rsmeans/partition/partition61.pdf
['Concrete block partition₩ncontrol joint₩nHorizontal joint reinforcing₩ngypsum plaster 2 coat₩nside finish₩n₩non masonry₩n₩n4₩n₩n1₩n₩n₩x0c', 'partition']
[['3 coat gypsum₩nmetal stud₩n₩nopposite face₩nExpansion joint₩n₩n2-1/2" @ 16" O.C.₩n3/8" gypsum lath₩nsame₩n₩n₩x0c', 'partition'], ['2 coat vermiculite₩nmetal stud₩n₩nopposite face₩nExpansion j
extraction finished :)
```

A-3 Text Classification Model

```
from google.colab import drive
drive.mount('/content/drive')

!pwd

%cd /content/drive/MyDrive/Colab\ Notebooks/Building\ Component\
Identification

!pwd
!ls
```

```
Mounted at /content/drive
/content
/content/drive/MyDrive/Colab Notebooks/Building Component Identification
/content/drive/MyDrive/Colab Notebooks/Building Component Identification
'barchart plot.ipynb'          'Multi with LSTM.ipynb'
 change_name.ipynb              partition
'drawing text.csv'             'PDF split'
'drawing text_integrated.csv'   rsmeans
'drawing text_revised.csv'      small_set
'drawing text_small_set.csv'   'text classification.ipynb'
'exterior wall'                'text classification_modified.ipynb'
 footing                       'Text extraction.ipynb'
 individual
```

```python
from collections import OrderedDict
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.layers import Dropout
import re
from nltk.corpus import stopwords
from nltk import word_tokenize
import nltk
from nltk.stem import PorterStemmer
nltk.download('stopwords')
STOPWORDS = set(stopwords.words('english'))
from bs4 import BeautifulSoup
!pip install chart-studio
from chart_studio import plotly as py
import plotly.graph_objs as go
import cufflinks
from IPython.core.interactiveshell import InteractiveShell
import plotly.figure_factory as ff
InteractiveShell.ast_node_interactivity = 'all'
from plotly.offline import iplot
cufflinks.go_offline()
cufflinks.set_config_file(world_readable=True, theme='pearl')
import random
```

```python
df = pd.read_csv('drawing text_integrated.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 314 entries, 0 to 313
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   text    314 non-null    object
 1   label   314 non-null    object
dtypes: object(2)
memory usage: 5.0+ KB
```

```
df['label'].value_counts()
```

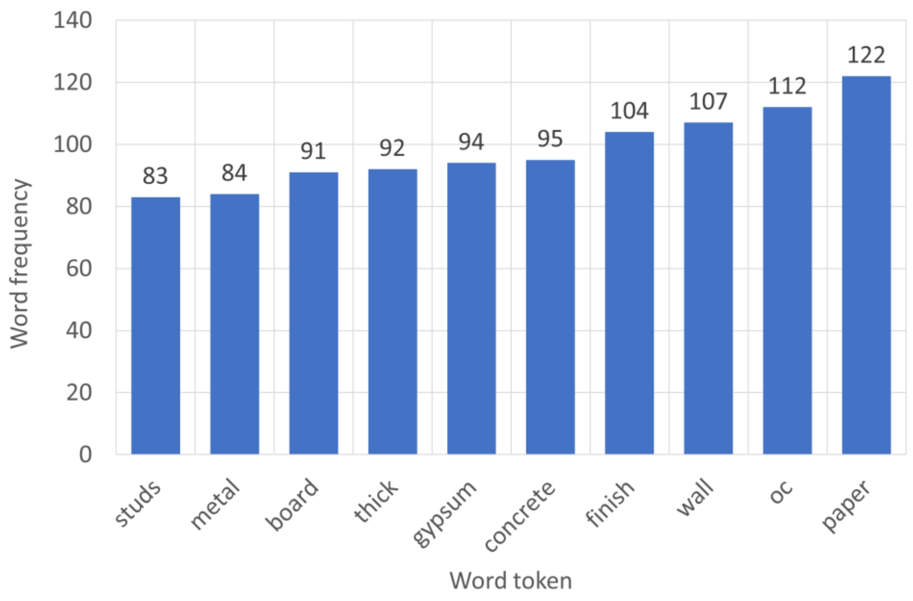
```
exterior wall    109
partition        103
footing          102
Name: label, dtype: int64
```

```
df = df.reset_index(drop=True)
REPLACE_BY_SPACE_RE = re.compile('[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile('[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))

def clean_text(text):
  text= text.lower()
  text = REPLACE_BY_SPACE_RE.sub('',text)
  text = BAD_SYMBOLS_RE.sub('', text)
  text = text.replace('x', '')
  text = ' '.join(word for word in text.split() if word not in S
TOPWORDS)
  return text
```

```
MAX_NB_WORDS = 122
tokenizer = Tokenizer(num_words=MAX_NB_WORDS, filters='!"#$%&()*+
,-./:;<=>?@[\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(df['text'].values)
word_index = tokenizer.word_index

print('Found %s unique tokens.' % len(word_index))
```

```
Found 803 unique tokens.
```

```
top10 = OrderedDict(sorted(tokenizer.word_counts.items(),key=lamb
da t : t[1])[-10:])

bar=plt.bar(top10.keys(),top10.values())
for rect in bar:
    height = rect.get_height()
    plt.text(rect.get_x() + rect.get_width()/2.0, height, '%.1f'
% height, ha='center', va='bottom', size = 12)
plt.xticks(rotation=45)
```



```
MAX_SEQ_LENGTH = 38
X = tokenizer.texts_to_sequences(df['text'].values)
X = pad_sequences(X, maxlen=MAX_SEQ_LENGTH)
print('Shape of data tensor:', X.shape)
```

```
Y = pd.get_dummies(df['label']).values
print('Shape of label tensor:', Y.shape)
```

```
Shape of label tensor: (314, 3)
```

```
TEST_SIZE = 0.4
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_siz
e = TEST_SIZE, random_state = 42)
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
```

```
LSTM_LAYER_NUM = 80
EMBEDDING_DIM = 60
DROPOUT_RATE = 0.2
random.seed(1)
model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length=X.s
hape[1]))
model.add(SpatialDropout1D(DROPOUT_RATE))
model.add(LSTM(LSTM_LAYER_NUM, dropout=DROPOUT_RATE, recurrent_dr
opout=0.2))
model.add(Dense(3, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
print(model.summary())
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_9 (Embedding)     (None, 38, 60)            7320      
                                                                 
 spatial_dropout1d_9 (Spatia  (None, 38, 60)           0         
 lDropout1D)                                                     
                                                                 
 lstm_9 (LSTM)               (None, 80)                45120     
                                                                 
 dense_9 (Dense)             (None, 3)                 243       
                                                                 
=================================================================
Total params: 52,683
Trainable params: 52,683
Non-trainable params: 0
_________________________________________________________________
None
```

```
EPOCH = 30
BATCH_SIZE = 15

VALIDATION_SPLIT = 0.2

history = model.fit(X_train, Y_train, epochs=epochs, batch_size=b
atch_size,validation_split=VALIDATION_SPLIT,callbacks=[EarlyStopp
ing(monitor='val_loss', patience=3, min_delta=0.0001)])
```

```
Epoch 1/30 10/10 [==============================] - 3s 103ms/step -
loss: 1.0845 - accuracy: 0.4267 - val_loss: 1.0718 - val_accuracy: 0.7368
Epoch 2/30 10/10 [==============================] - 1s 51ms/step -
loss: 1.0153 - accuracy: 0.7667 - val_loss: 0.9766 - val_accuracy: 0.7632
Epoch 3/30 10/10 [==============================] - 0s 50ms/step -
loss: 0.7952 - accuracy: 0.8067 - val_loss: 0.7723 - val_accuracy: 0.8158
Epoch 4/30 10/10 [==============================] - 1s 60ms/step -
loss: 0.5410 - accuracy: 0.8467 - val_loss: 0.5005 - val_accuracy: 0.8684
Epoch 5/30 10/10 [==============================] - 1s 58ms/step -
loss: 0.3556 - accuracy: 0.9067 - val_loss: 0.4144 - val_accuracy: 0.9211

                                  ⋮

Epoch 15/30 10/10 [==============================] - 1s 51ms/step -
loss: 0.1037 - accuracy: 0.9733 - val_loss: 0.1337 - val_accuracy: 0.9737
Epoch 16/30 10/10 [==============================] - 1s 53ms/step -
loss: 0.1110 - accuracy: 0.9667 - val_loss: 0.1238 - val_accuracy: 0.9737
Epoch 17/30 10/10 [==============================] - 0s 50ms/step -
loss: 0.0895 - accuracy: 0.9733 - val_loss: 0.1509 - val_accuracy: 0.9211
Epoch 18/30 10/10 [==============================] - 1s 52ms/step -
loss: 0.1010 - accuracy: 0.9600 - val_loss: 0.1440 - val_accuracy: 0.9211
Epoch 19/30 10/10 [==============================] - 1s 51ms/step -
loss: 0.0808 - accuracy: 0.9867 - val_loss: 0.1335 - val_accuracy: 0.9474
```

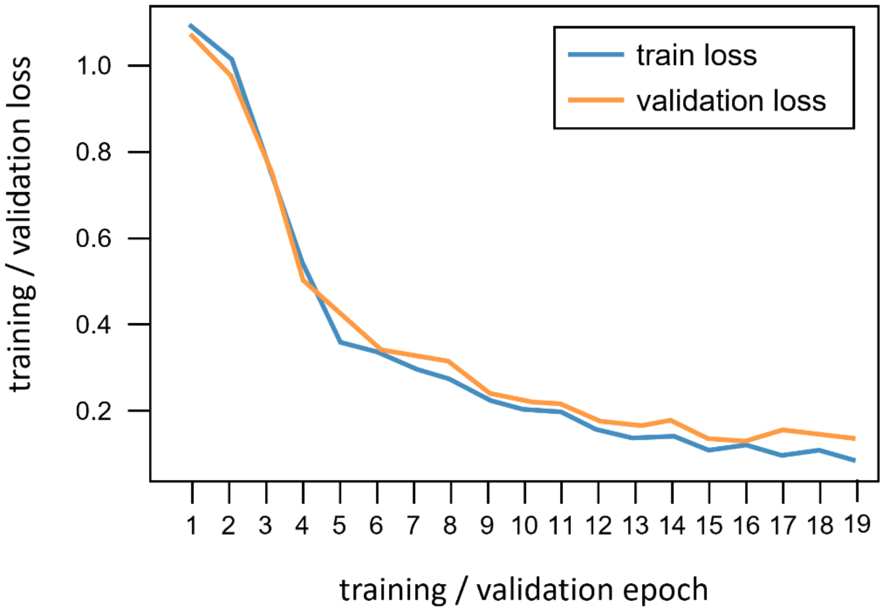
```
accr = model.evaluate(X_test,Y_test)
print('Test set\n  Loss: {:0.6f}\n  Accuracy: {:0.6f}'.format(acc
r[0],accr[1]))
```

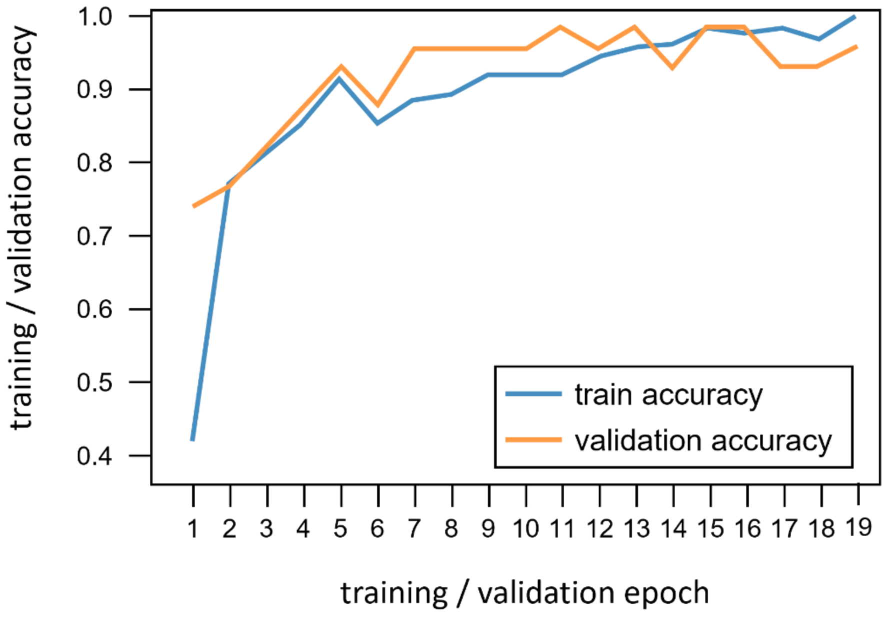
```
4/4 [==============================] - 0s 11ms/step - loss: 0.2292 - accuracy: 0.9524
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score

Y_hat_test = model.predict(X_test)
Y_hat_onezero = np.zeros_like(Y_hat_test)
Y_hat_onezero[np.arange(len(Y_hat_test)),Y_hat_test.argmax(1)] =
int(1)
Y_hat_onezero=Y_hat_onezero.astype(int)



Yacc_score = accuracy_score(Y_test,Y_hat_onezero)
Yprecision = precision_score(Y_test,Y_hat_onezero,average='macro'
)
Yrecall = recall_score(Y_test,Y_hat_onezero,average='macro')
Yf_1 = f1_score(Y_test,Y_hat_onezero,average='macro')
print("accuracy : %f \n"%Yacc_score)
print("precision : %f \n"%Yprecision)
print("recall : %f \n"%Yrecall)
print("f-1 : %f "%Yf_1)
```

```
accuracy : 0.952381

precision :0. 952328

recall : 0. 953398

f-1 : 0. 952231
```

```
plt.title('Loss')
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show();


plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()
plt.show();
```



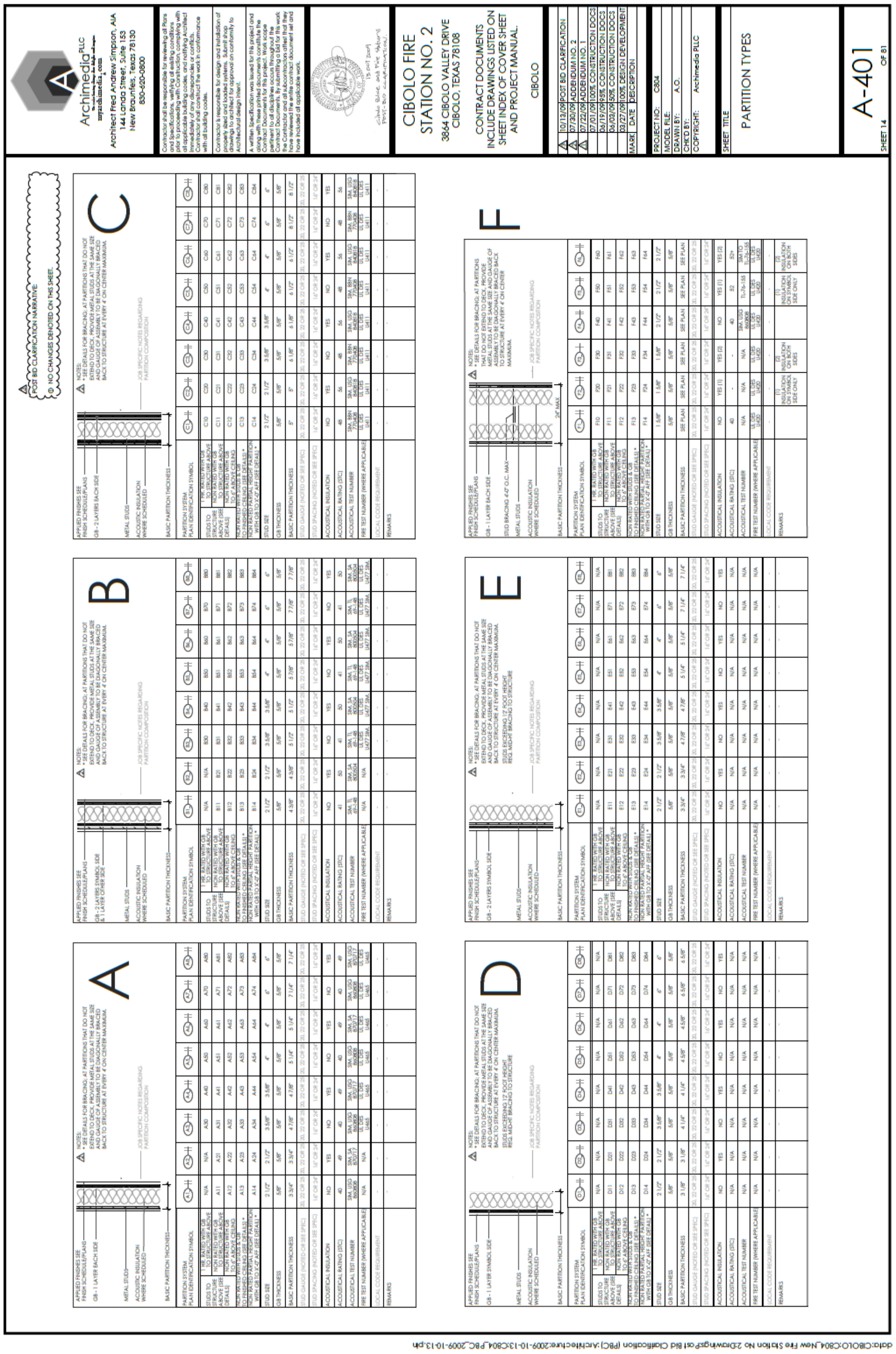

training / validation accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0.4
train accuracy
validation accuracy
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
training / validation epoch

APPENDIX B

SAMPLE DETAIL SECTION DRAWING

B-1 Partition Detail Section Drawings Sets from Cibolo Fire Station No.2

Architectural Plan Drawing Reprinted from Simpson (2009)


Archimedia PLLC
Architect Fred Andrew Simpson, AIA
144 Landa Street, Suite 153
New Braunfels, Texas 78130
830-620-0800
CIBOLO FIRE STATION NO. 2
3864 CIBOLO VALLEY DRIVE
CIBOLO, TEXAS 78108
CONTRACT DOCUMENTS
INCLUDE DRAWINGS LISTED ON SHEET INDEX OF COVER SHEET AND PROJECT MANUAL.
CIBOLO
10/13/09 POST BID CLARIFICATION
07/30/09 ADDENDUM NO. 2
07/22/09 ADDENDUM NO. 1
07/01/09 100% CONSTRUCTION DOCS
06/19/09 95% CONSTRUCTION DOCS
06/03/09 50% CONSTRUCTION DOCS
03/27/09 100% DESIGN DEVELOPMENT
MARK DATE DESCRIPTION
PROJECT NO: C804
DRAWN BY: A.O.
COPYRIGHT: Archimedia PLLC
SHEET TITLE
PARTITION TYPES
A-401
SHEET 14 OF 81
POST BID CLARIFICATION NARRATIVE:
NO CHANGES DENOTED ON THIS SHEET.
A
B
C
D
E
F
data:CIBOLO:C804_New Fire Station No 2:Drawings:Post Bid Clarification (PBC):Architecture:2009-10-13:C804_PBC_2009-10-13.pln