Towards a Comprehensive Façade Inspection Process: An NLP based Analysis of Historical Façade Inspection Reports for Knowledge Discovery

Zhuoya Shi, Keundeok Park, and Semiha Ergan

Tandon School of Engineering, New York University, the USA
E-mail: zs1110@nyu.edu, kp2393@nyu.edu, semiha@nyu.edu

Abstract –

Façade condition assessment for buildings is essential to public safety in cities. Currently, twelve major cities across the U.S. ensure the building façade safety with mandatory façade inspection programs. Even in major cities with the façade inspection programs, there have been seventeen falling debris accidents reported in 2019, three of which were fatal. These accidents indicate a need to improve the current façade inspection practice. Shadowing work conducted by the research team with expert inspectors on three buildings, and analysis of façade inspection programs and guidelines show that inspectors check façades based on defect types or on façade components, whereas existing documentation to guide inspectors are based on major material types. This mismatch results in inspectors checking façade components based on their experience, which might not align well with the expectations of agencies. Systematic and detailed assessment guidance is necessary to get a comprehensive and consistent façade inspection. Towards such systematic guidance and to understand the underlying reasons for continuing accidents, this paper provides the details of an approach to identify generic vocabularies and the relationships between major entities that play a role in the inspection domain for systematic inspection processes. To identify these, we developed a data-driven approach that analyzed around 100 façade inspection reports that were filed to the NYC Department of Buildings (DOB) during the past inspection cycle (2014-2019). Among the twelve major cities, New York City (NYC) has the longest history of façade inspection, and most buildings (14,000) enrolled in the façade inspection program. We believe that study about NYC buildings can provide a general understanding of inspection requirements in other cities where similar problems exist. The developed mechanism is based on natural language processing and unsupervised machine learning techniques and is used to extract the vocabularies of façade elements, defect types, associated defect attributes, and mapping between them. The results also provide the mapping relationship of façade components and defect types for a specific façade type (e.g., stone/limestone). This work provides the foundation for an ontology to be used to systematically guide façade inspection for any given building.

Keywords –

Façade inspection, Report analysis; Natural Language Processing (NLP); Unsupervised learning.

1 Introduction

Twelve major cities in the U.S. have a façade inspection program to ensure the safety of buildings. Accidents and incidents caused by debris falling from building façades, however, are still happening in cities. For example, in the past year (2019), 17 accidents due to debris falling occurred in the U.S., causing deaths and injuries. The past decade has shown that more than 700 complaints about façade safety were received annually by the department of buildings (DOB) in the city of New York (NYC) [1]. These accidents and complaints indicate a need to improve the current façade inspection practice.

Shadowing of façade inspectors by the authors resulted in the identification of several challenges in the current façade inspection practice [2]. One of these challenges is the lack of a comprehensive and detailed checklist that can serve as a guideline to a systematic inspection that is based on defect and façade component types instead of material types. Current façade inspection regulations in different cities and international guide for standard periodic façade inspections published by the standards organization [3] cover information on conditions of buildings that need an inspection, length of inspection cycles, general and close visual inspection of façade, and reports formulation and submission. Information about what defect types need to be checked and what associated attributes are essential for façade safety assessment, however, is missing from the

regulations and practice standard documents. Aside from the regulations and practice standards, a few glossaries formulated by industry professionals aim to provide descriptions and illustrations of defect types for different façade materials [4-6]. The glossaries are only for educational purposes and are not integrated into any guideline for defect identification in the façade inspection practice. Façade inspection is still a practice that relies solely on the personal experience of each inspector and company-specific templates. Figure 1 shows an example of different inspection results in two inspection reports for the same brick masonry building and its roof.

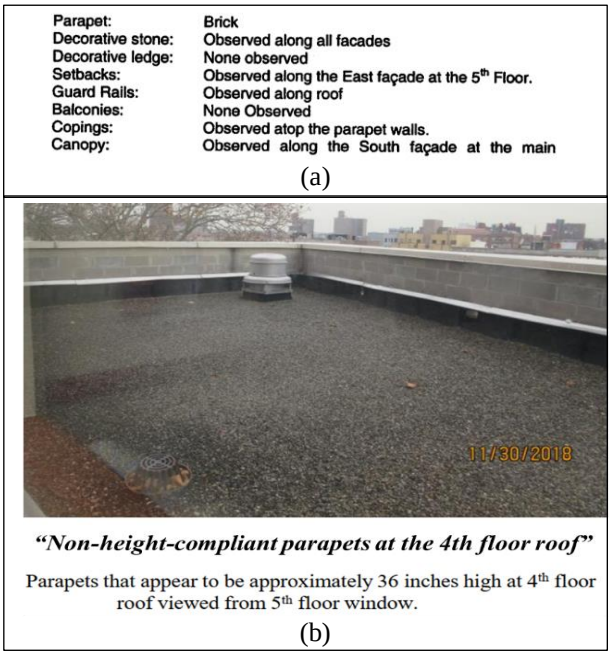

Figure 1. Examples of different inspection results for the same roof. (a) Previous inspector (Cycle 7) only mentioned the existence of a parapet. (b) The current inspector (Cycle 8) checked the height of the parapet with respect to the code compliance.

For safety concerns, the height of parapets should comply with the construction code that was applicable when the building was constructed. The inspector, who conducted the inspection work in the prior cycle, only mentioned the existence of a parapet at the roof (Figure 1a) and its material type as the information collected for the parapet. The inspector we shadowed, however, also checked the height of the parapet, compared it to the requirements in the code, and identified that the height of the parapet is not compliant with the applicable building code (Figure 1b). Such differences of what components to check, what specific parameters to check for each façade component are common, and show a need for a checklist that the inspectors can follow to conduct an in-depth and comprehensive inspection regardless of their experience.

Shadowing work and the review of submitted reports showed that the inspection is conducted per façade component and is not based on material types. For example, when an inspector is checking a balcony, he/she will check the concrete panels for cracks, the railings for connection stability, and compliance of spacing between railings and their heights to the code altogether, instead of checking cracks in all concrete components at once. One example of this component-based practice is provided in Figure 2. The inspector, who submitted the report, grouped the observed conditions based on the façade components (i.e., openings). The current façade condition glossaries, however, summarize the defect types based on façade materials (e.g., concrete, metal) and there is a need to know which defects are applicable to which façade components for a comprehensive and consistent façade inspection.

Masonry Openings *a.* ***Window Frames*** *Description: The windows are constructed of aluminum. Condition: The windows were observed to be in satisfactory condition during the critical examination (see photo #13).Classification: SAFE b.* ***Window Sills*** *Description: The windowsills are constructed of brick masonry. Condition: The windowsills were observed to be in satisfactory condition during the critical examination. Classification: SAFE*

Figure 2. Façade inspection report grouping the conditions based on façade components.

Towards a comprehensive façade inspection guidance, what is needed is the knowledge of how defect types (and defect-related attributes) map to façade component types, and the comprehensive vocabularies for these defect types, defect attributes, and façade component types. The objective this paper is to identify this mapping and the vocabularies of the defect, attribute, and façade component types. With this objective, we developed an approach using bidirectional long short-term memory (bi-LSTM) model to automatically analyze around 100 façade inspection reports that were submitted in the past inspection cycle (2014-2019) for buildings in NYC. Here, the assumption is that inspectors' practice is reflected on the inspection reports they submit and can be leveraged for investigating the relationships between façade components and defects, and expanding the available material-specific vocabularies towards component-specific vocabularies. This paper provides an overview of the developed bi-LSTM model and the results of this mapping and resulting vocabularies of façade components, defect types, and defect attributes. These findings also lay the foundation for an ontology for systematic and comprehensive façade inspection.

2 Literature Review

Here we present a synthesis of previous researches on urban façade inspection, structural health monitoring, and the application of natural language processing (NLP) in the civil engineering domain.

Previous researches on façade inspection and structural health monitoring were mainly focused on automatic defect detection using digital data, particularly point clouds and images. Researchers developed algorithms to detect cracks [7-9], moisture [10], spalling [8, 9], vegetation [8], and efflorescence [9] on concrete, masonry, and steel surfaces. Several algorithms are able to quantify the defects, such as the area of spalling [11] and the width of cracks [7] automatically or manually on 3D models [12]. Although these studies provide a handful of defect types/attributes on a select type of materials, their focus is not aligned with this study that aims to define the vocabularies of defects, component types, and the mapping between them for improvement of the façade inspection practice.

Previous NLP applications in the civil engineering domain aim to extract essential information such as condition assessment information from bridge inspection reports, or project relevant information from unstructured construction documents, or regulatory information from building codes with different machine learning models [13-15]. Although there are NLP applications in the domain, the analyzed documents are semi-structured or structured with logical-connected words (e.g., "*equal to*" and "*less than*" represent quantification compliance in the building codes) or with known defect types (e.g., crack, spall, efflorescence, etc.) and rating categories (e.g., minor, moderate, severe) [16], which can be analyzed with rule-based information pattern matching. Different from previously studied document types, façade inspection reports are personal narrative descriptions written without guidance or specific templates. Thus, this unstructured nature of the inspection reports requires unsupervised learning to find relevant words (e.g., parapet, crack, diagonal), label them correctly (e.g., façade component, defect type, defect attribute) within the context of a report, and maintain the relationships between these labels. We implemented a natural language processing (NLP) algorithm to identify the vocabularies for façade components, defect types, defect attributes, and the hidden relationships between façade components and defect types. Long short term memory network (LSTM) is one of the Recurrent Neural Network (RNN) architectures and is known to be suitable for processing sequence data (e.g., time-series, speech sequence, and text) because LSTM preserves information about the data that the model sees earlier in a sequence. Unlike general sequential data processing, NLP has a feature that the context in a sentence is defined by the interrelationships among words in both backward and forward directions. Since the unidirectional LSTM passes input information forward only, bidirectional LSTM is more proper for automated understanding of natural language. In addition, the conditional random field (CRF) is a statistical modeling method that is widely known for its high performance in predicting a label for sequence data. Adding CRF as an output layer in the bi-LSTM model (i.e., bi-LSTM-CRF model) is proved to outperform previous state-of-the-art models in the information labeling tasks [17]. It has been utilized in the approach presented in this paper. To further improve the performance of NLP models, skip-gram models, which are unsupervised learning models and can precisely capture the semantic (i.e., meaning) and syntactic (i.e., grammatical structure) relationships between words [18]in a context. Previous research studies trained skip-gram models with general English text and used them to improve the performance of machine learning models on extracting information from documents (e.g., [13]). In this study, we trained a skip-gram model with domain-specific text (i.e., façade inspection reports text). We used it to generate the dependency information that was included in the input of the machine learning model to boost the performance of the model.

3 Research Method

To obtain comprehensive vocabularies for façade components, defect types, and their attributes together with the mapping relationship among those, we developed an algorithm that is capable of extracting key information in unstructured inspection reports. In a nutshell, this algorithm includes a trained bi-directional long short term memory (bi-LSTM) model, boosted with outputs of an unsupervised skip-gram model developed by the research team to capture semantic and syntactic relationships between words (as detailed in section 4).

For the training of the bi-LSTM models that we enhanced with domain-specific skip-gram models, first, we needed to generate initial vocabularies of façade components, defect types, and defect attributes. For this purpose, we reviewed current façade inspection practice standards [3], local laws/regulations [19], and façade condition glossaries [4-6] to generate defect types and attributes vocabularies. Similarly, we reviewed building component libraries of 3D modeling tools (e.g., Revit libraries) and building classification models (e.g., Uniformat) to prepare the façade components vocabulary. We defined four labels, which are ***façade-component***, ***defect-type***, ***defect-attribute***, and ***others*** (i.e., information that is irrelevant to façade conditions), to automatically label worlds in reports. The details of this approach are provided in the next section.

4 Boosted Bi-LSTM model for mining of inspection reports

Figure 3 provides an overview of our approach. Data preprocessing (step 1) and labeling (step 2) steps are provided in section 4.1, dependency information provided by the skip-gram model (step 3) is presented in section 4.2, and the utilization of the model to identify vocabularies and mapping relationships (step 4) is discussed in section 4.3.

4.1 Preprocessing and labeling of the text

We obtained around 3,000 façade inspection reports that cover all the seven façade types in document format from the NYC DOB database. For this study, we analyzed 100 inspection reports for only stone limestone façade buildings. Preprocessing starts with using a parser to extract the raw text from the sections where inspectors describe their findings and splitting the raw text into indexed sentences with the sentence tokenizer. The indexed sentences were tokenized into words (i.e., separating a given sentence into the list of its words), then lemmatized (i.e., returning to the root form of the words) to eliminate the influence of different tenses, plural forms, verb/noun differences. The part-of-speech (POS) information of each token (e.g., noun, verb, adverb, adjective, etc.) was used to achieve a more accurate lemmatization. For instance, the word "*rose*" can be the past tense of "*rise*" or the flower. Depending on the POS of the word, the result of lemmatization differs. These lemmatized sentences were temporarily used in the labeling process to eliminate labeling errors due to the different formats of the words in the text and the identified vocabulary list. Next, we labeled the preprocessed and lemmatized text using the Inside-outside-beginning (IOB) tagging mechanism. With this tagging mechanism, we are able to label compound words that are consisted of multiple words with the beginning token (i.e., a word in this context) and inside tokens. The "**I-**" tag stands for "inside" of a label, referring to the subsequent word in a compound word (e.g., "*escape*" in "*fire escape*"). The "**O**" tag stands for "outside" of a label, referring to the irrelevant word. The "**B-**" tag stands for "beginning" of a label, referring to the first word in a compound word (e.g., "*fire*" in "*fire escape*"). For example, in the term "the window lintel", "window lintel" should be labeled as a single ***façade-component***. Using IOB tagging mechanism, the word "*the*" is labeled as "**O**", and "*window lintel*" will be automatically labeled as [window (B: façade-component) lintel (I: façade-component)]. The label categories we used are façade components, defect types, defect attributes (i.e., length, width/depth, direction, location, material, number of floors, and façade sections), and others.

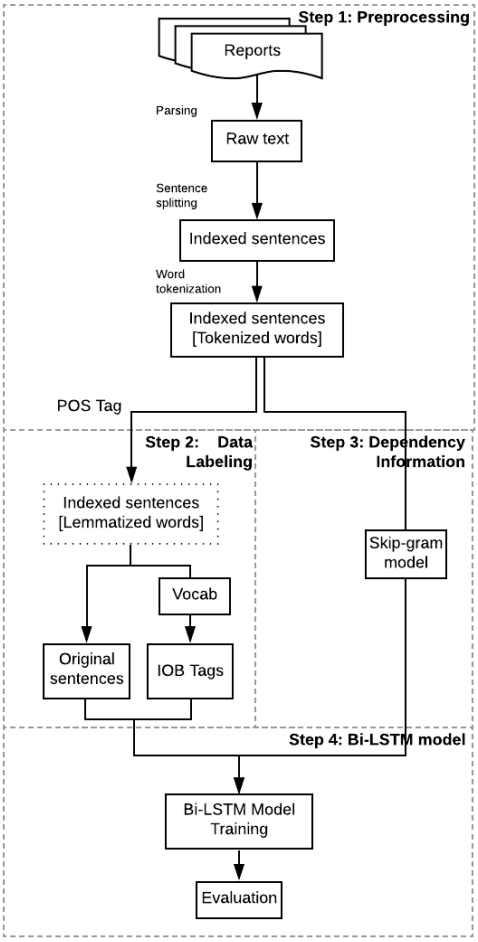

Figure 3. Overview of the developed approach.

Figure 4 provides an example of a sentence from the reports and its labeling result. If there are no compound words, each identified word that corresponds to one of the four label categories will be labeled with "**B**". All the other words that are not of interest in this labeling process are to be labeled as "**O**" (i.e., others).

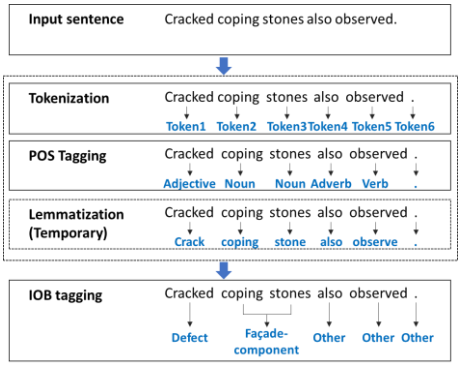

Figure 4. An example of a sentence preprocessed and labeled with the IOB tagging mechanism.

4.2 Obtaining dependency information through the domain-specific skip-gram model

Word embedding is the process of vectorization of words to convert natural language into a computer-readable format based on the extracted features of the language. Word embedding approach takes a large input of tokens, projects them to a vector space, and assigns each unique token a corresponding vector for the representation in the vector space. In this study, we used two word embedding approaches to compare the results of the bi-LSTM model with and without dependency information. These two embedding approaches are the one-hot encoding, which does not provide dependency information between words, and the skip-gram model, which includes the dependency information in the word embedding process. One-hot encoding is a vector representation of word in binary vector format (e.g., *crack = [1,0,0], window = [0,1,0], maintenance = [0,0,1]*). Since the vectors represent words in binary value, one-hot encoding cannot provide the similarity/distance between words. On the other hand, the skip-gram model is an unsupervised learning method for word embedding that uses vector space to capture and represent the semantic and syntactic relationships between words efficiently. Semantic refers to the word meanings and relations, and syntactic refers to the rules of a language and grammatical arrangement of words in sentences. For example, sentences like "*window identifies crack*" ("subject"+ "verb" + "adjective") is a syntactically correct sentence, but it is not semantically correct. After the word embedding, all the input words are mapped into the N-dimensional vector space ($word_1 = [v_1, v_2, v_3, v_4, \ldots, v_n]$). Since the mapped words are using vector representation (e.g., word "*crack*" is represented as [-0.478, 0.917, 0.196, -0.443, -0.978, …, -0.207]) with respect to the other words in the same context, the skip-gram model can perform analogical reasoning precisely. In other words, if two words are placed at similar locations in two different sentences, and the meaning of the terms are similar, then the distance between those two words is relatively small in the vector space if the same pattern is observed often by the skip-gram model. For example, given the following two sentences from a report: (1) "*We recommend repairs to be made to correct these deficiencies by February*." (2) "*We recommend timely maintenance to repair the SWAMP condition observed our inspection*." In vector space, the distance between "*repairs*" and "*maintenance*" is relatively small. Figure 5 presents an example for a simplified 2D projection of the vector space showing the distance between "*repairs*" and "*maintenance*".

Previous studies indicate that the skip-gram model trained with general English text improves the information labeling model performance [13]. In this study, we trained a skip-gram model with the façade inspection reports and included the dependency information, which is represented by vectors generated from the skip-gram model, as part of the input for the bi-LSTM model.

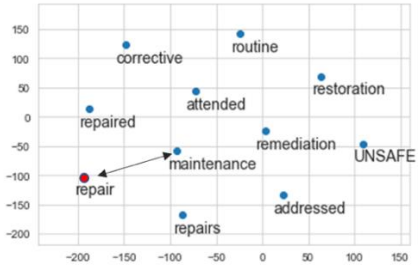

Figure 5. An example of the tokens "repair" and "maintenance" in projected vector space.

4.3 Identifying vocabularies with the boosted bi-LSTM-CRF model

In this study, we built and trained a bi-LSTM-CRF model boosted with skip-gram model outputs. The input for the bi-LSTM-CRF model is the labeled sentences (i.e., the output of step 2 in Figure 3) and the dependency information (i.e., the output of step 3 in Figure 3). Figure 6 provides an illustration of the boosted bi-LSTM-CRF model with a short sentence as an example. The model has an input layer, where the preprocessed sentence "*Exterior wall has crack*." is converted into vector representation by word embedding as input into the bi-LSTM layers. Then, the LSTM model learns from the input data in both forward and backward propagations to update the parameters. The output of bi-LSTM is then inputted to the CRF layer to refine the relational information between tags. The CRF layer works as a classifier to predict the label of each token and improves the performance of the model because it can learn the constraints of labels based on their positions. Constraints such as "inside labels (i.e., I) cannot exist without beginning labels (i.e., B)", and "I of one label cannot appear after B of another label (e.g., 'I-defect' cannot appear after 'B-component')" are learned by this layer. After the CRF layer, the final output of this bi-LSTM-CRF model is the labels for each token in the sentence.

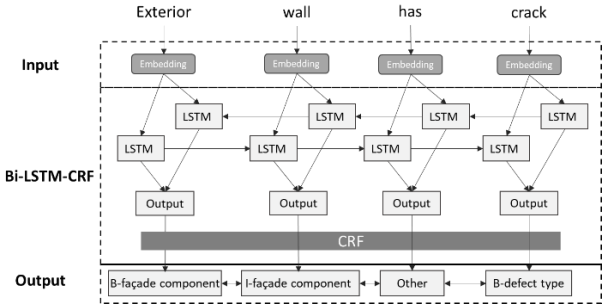

Figure 6. Illustration of bi-LSTM-CRF models with an example sentence.

4.4 Model development

The sentences extracted from the reports were randomly separated into training (i.e., data that is used to build the models), validation (i.e., data that is used to prevent overfitting problem in the training process), and testing (i.e., data that is used to test the performance of the models) sets with 7:2:1 ratio. We trained two bi-LSTM models with one of them included the dependency information provided by the skip-gram model and the other one using one-hot code input. We trained the models with the training set, and the validation set was used to fine-tune the models and avoid the overfitting problem. The testing set remains unseen by the model and is used for model performance evaluation. After we trained and optimized the models, the sentences in the testing set are labeled by the model. The output labels are compared with the labels provided by the IOB tagging mechanism. We calculated the F_1 score, precision, and recall for model performance evaluation using Equations 1-3, where TP represents true positive (i.e., the number of tokens that were correctly labeled), FP represents false positive (i.e., the number of tokens that were mislabeled), and FN represents false negative (i.e., the number of tokens that should be labeled but were missed). The evaluation metrics can be calculated with the following rules. Precision is the ratio of correctly labeled tokens to all the tokens that are labeled in the testing set for each label (Equation 1). The recall is the ratio of the correctly labeled tokens to the total number of correct labels expected to be labeled in the testing set for that label (Equation 2). F_1 score is the weighted average of precision and recall (Equation 3). The performances of both models are evaluated using stone/limestone façades.

$$Precision = \frac{TP}{TP + FP} \quad \text{Equation(1)}$$

$$Recall = \frac{TP}{TP + FN} \quad \text{Equation(2)}$$

$$F_1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \text{Equation(3)}$$

5 Preliminary results and discussions

This section provides the results of the labeling process using the developed boosted bi-LSTM-CRF model. The results are provided for reports analysis of stone/limestone façades.

5.1 Preliminary results of vocabulary identification in stone/limestone façade inspection reports

For stone/limestone façade buildings, 100 façade inspection reports were available, and 1771 sentences were extracted from the reports. The training, validation, and testing sets had 1272, 319, and 180 sentences, respectively. After preprocessing and IOB tagging, there were 3446 tokens labeled as façade components, 190 tokens labeled as defect types, and 256 tokens as defect attributes in the training set. Table 1 shows the precision, recall, and F_1 score for each label category identified by the developed approach. As shown in Table 1, the left columns with normal text show the model performance when the input data does not include dependency information. The right columns with bold text show the model performance when we include the dependency information as part of the input. The model performs well in the vocabulary labeling for façade components, defect types, and defect attributes given their consistently high scores in precision and recall. The labeling performance for both façade components and defect types has been improved approximately by 10% by including the dependency information, while the labeling performance for defect attributes did not change much by including the dependency information. Resulting vocabularies after the trained model is finalized for stone/limestone façade are discussed with respect to their categories.

Vocabulary for façade component types: For façade component types, a total of 38 component types (e.g., cornice, storefront, column, mortar joint, coping stone, etc.) were identified for stone/limestone façades as a result of this automated report analysis. All the identified façade component types were covered by our initial vocabulary, and no additional façade components were expected to be identified.

Table 1. Precision, recall, and F-1 score for labeling (with/without dependency information).

Label category	Precision		Recall		F-1 score	
Dependency information	w/	**w**	w/	**w**	w/	**w**
Façade component	0.78	**0.88**	0.85	**0.91**	0.82	**0.90**
Defect type	0.62	**0.76**	0.59	**0.74**	0.61	**0.75**
Defect attributes	0.91	**0.91**	0.74	**0.77**	0.82	**0.83**

Bold w: model performance with word embedding information. **w**: with; w/: without.

Vocabulary for defect types: There were 12 defect

types identified by the automated report analysis of stone/limestone façade inspection reports. These include cracking, peeling, missing, hazardous condition, removed, spalling, corrosion, displacement, bulging, delamination, chipping, and hollow. Certain defects such as deformation, misalignment, and loose that are important to the safety condition assessment were not mentioned in the reports.

Vocabulary for defect attributes: The defect attributes included in the inspection reports of stone limestone buildings are location (e.g., corner, roofline, floor line, joint, etc.), direction (e.g., horizontal, vertical, inward, etc.), material (e.g., paint, rust, etc.), section (i.e., east/west/north/south façade), area (e.g., square feet), and related deterioration (e.g., chalking, staining, discoloration, etc.). Among all the defect attributes, related deteriorations, and façade section information was not identified correctly by the approach.

5.2 Mapping relationship between defect types and façade components for stone/limestone façades

An important outcome of the trained models is the discovery of the unknown mapping relationships between façade components and defect types, which are required for a systematic component-based façade inspection guidance. The mapping relationships identified for the stone limestone façades are provided in Figure 7. Numbers in each cell indicate the frequency of the "façade component-defect type" pairs that appeared in the façade inspection reports. The frequency of identified pairs is normalized over the defect types to show the most frequent defect types on each façade component. The density of color in each cell represents the normalized results. The underlying assumption here is that the vocabularies appeared in the same sentence is describing one defect observed on a façade component by the inspector. For stone/limestone façades, 38 building elements and 12 defect types are identified from the façade inspection reports. When the figure is analyzed vertically, it is possible to identify the relations discovered between façade elements and a given defect type. Crack is the defect type that relates to most of the façade elements to be inspected for stone/limestone façades and is related with 30 out of 38 façade elements. Hazardous condition (associated with 15 façade elements), spalling (associated with 14 façade elements), and peeling (associated with 10 façade elements), are defect types that rank high in mapping to façade components. Delamination and chipping, both associated with 6 façade elements, are defect types that rank low in mapping to façade components. Hollowness, which is only associated with coping, is the defect type that relates to the least number of façade components.

When Figure 7 is analyzed horizontally, it is possible to observe what types of defects are applicable to a given façade component. Among all 38 façade elements, brick pieces (that require inspection for 11 different defect types), copings (that require inspection for 10 different defect types), parapets (that require inspection for 9 different defect types), and lintels (that require inspection for 8 different defect types) are the façade components that have the highest number of defect types associated with them. On the other hand, glass panels, columns, window panels, ladders, entrance doors, and fascia are the façade elements that only have crack associated with them.

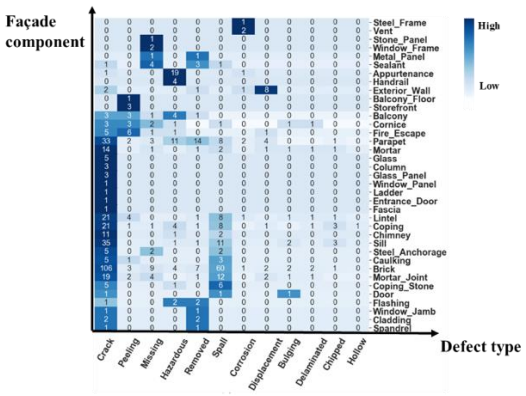

Figure 7. Mapping of façade components and defect types for stone/limestone façades

6 Conclusions

In this study, the authors proposed an automated approach for analyzing façade inspection reports to extract vocabularies for critical façade inspection information (e.g., defect type, façade component types) and discover the undocumented but critical information on how defect types map to façade component types. This information is needed to streamline the façade inspection process by providing the flexibility for inspectors to organize their inspection findings per façade component type or per defect type. A natural language processing approach, combined with an unsupervised skip-gram model, was developed and trained using NYC façade inspection reports. By comparing the labeling results, we proved that including the dependency information can improve the performance of bi-LSTM-CRF in information identification and labeling. NYC has the longest history of the façade inspection program and the largest number of buildings that need inspection regularly-hence it has been used as a test site. The analysis conducted in this study can shed light on other cities where an in-depth understanding of the façade inspection practice is needed. As an immediately following work, the authors will analyze the façade inspection reports of buildings with other façade types

(e.g., brick masonry, metal panel) and include a larger set of reports to improve the performance of the approach. The vocabularies developed and the mappings identified in this study can be helpful for future research that aims to provide a model-based comprehensive guide for the façade inspection practice.

References

[1] NYC OpenData. DOB Complaints Received. Online: https://data.cityofnewyork.us/Housing-Development/DOB-Complaints-Received/eabe-havv, Accessed: 06/05/2020

[2] Shi Z. and Ergan S. Towards point cloud and model-based urban façade inspection: challenges in the urban façade inspection process. In *Proceedings of the ASCE Construction Research Congress*, Tempe, Arizona, 2020.

[3] ASTM International. Standard guide for conducting inspections of building facades for unsafe conditions. Online: https://compass.astm.org/EDIT/html_annot.cgi?E2841+19, Accessed: 06/05/2020.

[4] Eschenasy D. Façade conditions: an illustrated glossary of visual symptoms. Online: https://standardwaterproofing.com/wp-content/uploads/2016/09/FacadePresentation.pdf, Accessed: 06/05/2020.

[5] Vergès-Belmin V. ICOMOS-ISCS: illustrated glossary on stone deterioration patterns. Online: https://www.icomos.org/publications/monuments_and_sites/15/pdf/Monuments_and_Sites_15_ISCS_Glossary_Stone.pdf, Accessed: 06/05/2020.

[6] Kopelson E. Conditions glossary. Online: https://vertical-access.com/resources/conditions-glossary/, Accessed: 06/05/2020.

[7] Anil, E. B., Akinci, B., Garrett, J. H., and Kurc, O. Characterization of laser scanners for detecting cracks for post-earthquake damage inspection. In *Proceedings of the International Symposium on Automation and Robotics in Construction*, Montreal, Canada, 2013.

[8] Brackenbury, D., Brilakis, I., and DeJong, M. (2019). Automated Defect Detection For Masonry Arch Bridges. In *International Conference on Smart Infrastructure and Construction 2019 (ICSIC) Driving data-informed decision-making*, pages: 3-9, Cambridge, UK, 2019.

[9] Wang, N., Zhao, Q., Li, S., Zhao, X., and Zhao, P. Damage classification for masonry historic structures using convolutional neural networks based on still images. *Computer‐Aided Civil and Infrastructure Engineering*, 33(12): 1073-1089, 2018.

[10] Feng, C., Liu, M. Y., Kao, C. C., and Lee, T. Y. Deep active learning for civil infrastructure defect detection and classification. In *Computing in civil engineering 2017*, pages: 298-306, Seattle, Washington, 2017.

[11] Kim, M. K., Sohn, H., and Chang, C. C. Localization and quantification of concrete spalling defects using terrestrial laser scanning. *Journal of Computing in Civil Engineering*, 29(6): 04014086, 2015.

[12] Russo, M., Carnevali, L., Russo, V., Savastano, D., and Taddia, Y. Modeling and deterioration mapping of façades in historical urban context by close-range ultra-lightweight UAVs photogrammetry. *International Journal of Architectural Heritage*, 13(4): 549-568, 2019.

[13] Li, T., and Harris, D. Automated construction of bridge condition inventory using natural language processing and historical inspection reports. In *SPIE Smart Structures + Nondestructive Evaluation*, Denver, Colorado, 2019.

[14] Fan, H., Xue, F., and Li, H. Project-based as-needed information retrieval from unstructured AEC documents. *Journal of Management in Engineering*, 31(1): A4014012, 2015.

[15] Zhang, J., and El-Gohary, N. M. Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking. *Journal of Computing in Civil Engineering*, 30(2): 04015014, 2016.

[16] Hartle, R. A., Ryan, T. W., Mann, E., Danovich, L. J., Sosko, W. B., Bouscher, J. W., and Baker Jr, M. Bridge Inspector's Reference Manual: Volume 1 and Volume 2 (No. DTFH61-97-D-00025). United States, Federal Highway Administration, 2002.

[17] Huang, Z., Xu, W., and Yu, K. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991, 2015.

[18] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages: 3111-3119, Lake Tahoe, Nevada, 2013.

[19] NYC DOB. NYC Construction Codes §28-302.1. Online: https://www1.nyc.gov/assets/buildings/building_code/2008_cc_ac_combined.pdf, Accessed: 06/05/2020.