Cloze: A Building Metadata Model Generation System based on Information Extraction

Fang He
The Hong Kong Polytechnic University
fangf.he@connect.polyu.hk

Dan Wang
The Hong Kong Polytechnic University
dan.wang@polyu.edu.hk

ABSTRACT

Recently, we have seen a flourish of data-driven building applications. It has also been noted that the main effort in application development today is on data preprocessing. More specifically, buildings have entities, e.g., a chiller. To extract their data values or to control the entities, applications need to refer to the *metadata* of a building, i.e., the data describe the entities in a building. Data preprocessing organizes the raw metadata of a building into a form that can be easily recognized by applications. Different buildings have different metadata conventions. Data preprocessing today is largely an ad hoc process and is manually done in a building-by-building manner.

How to automate data preprocessing is challenging. In this paper, we first formulate a problem on converting building raw metadata with ad hoc conventions into a building metadata model that follows a standard convention, e.g., the Brick metadata schema. Depending on application scenarios, we present three variants of the problem. This problem is intrinsically a text analysis problem. We thus propose to leverage the information extraction paradigm, a type of document processing to extract structured information from unstructured documents/texts. We analyze real-world building metadata and present a set of challenges on corpus denoise, coreference resolution, disambiguity, etc. We develop a system, Cloze with corresponding solutions. We evaluate Cloze with six real-world buildings. Our results show that Cloze can learn and automatically recognize raw metadata and their relations with an accuracy of 96.3%.

CCS CONCEPTS

• **Information systems → Data Management System**.

KEYWORDS

Smart building, Data Model, Metadata, Information Extraction

ACM Reference Format:
Fang He and Dan Wang. 2022. Cloze: A Building Metadata Model Generation System based on Information Extraction. In *The 9th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation (BuildSys '22), November 9–10, 2022, Boston, MA, USA*, 10 pages. https://doi.org/10.1145/3563357.3564066

BuildSys '22, November 9–10, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9890-9/22/11…$15.00
https://doi.org/10.1145/3563357.3564066

1 INTRODUCTION

In recent years, we have seen a flourish of data-driven building applications [3, 30, 35]. It has also been noted that the main effort in developing such applications today is on data preprocessing of the raw metadata of buildings [10]. Buildings have abundant raw *metadata*, i.e., the data that describe the entities in buildings. For example, the metadata "WKGO-CP01" refers to a chiller in the WKGO building with ID 01. A piece of metadata is used to pinpoint an entity so as to extract the data values of this entity or to control this entity. Raw metadata should be preprocessed into an organization so that applications can easily recognize them. For example, WKGO-CP01 should be labeled as a chiller. Different buildings have different metadata conventions, e.g., in another building, WTS_Chiller refers to a chiller. Unfortunately, these conventions are only human-readable. Thus, data preprocessing today is largely an ad hoc process and is manually done in a building-by-building manner.

To support data-driven building applications, *building metadata schema* have been developed [1, 7]. A building metadata schema is a predefined organization of building metadata. One notable example is Brick [7], where the metadata should be organized into a triple (entity, relation, entity), e.g., (chiller, hasLocation, room). The Brick schema defines a set of entity classes and relation classes commonly used in buildings (e.g., a Chiller entity class and a hasLocation relation class). Building applications can be developed on top of a standard building metadata schema to easily pinpoint building entities. Nevertheless, the building metadata of a specific building are developed with its own conventions to satisfy its own needs, and they may not match a standard building metadata schema such as Brick. It is still a manual process to develop the *building metadata model* that can follow a building metadata schema for a specific target building. How to automate this process is challenging.

In this paper, we study the *building metadata model generation problem* and we study it in the context of the Brick schema. We note that, in practice, there are source buildings with labeled metadata that can be used for training. We develop and train machine learning (ML) models to classify building metadata into Brick entity classes and the building metadata pairs into Brick relation classes. For example, metadata WKGO-CP01 should be classified into a Chiller entity and WKGO-BF into a Basement Floor entity; and the pair WKGO-CP01 and WKGO-BF into a hasLocation relation. We present three variants of the problem: (P1) there is no labeled data for the target building, i.e., only labeled data from source buildings; (P2) there is no labeled data for the target building but the target building has specification files to describe its metadata convention; and (P3) there are (partial) labeled data from the target building. Existing studies, e.g., Scrabble [13] and ProgSyn [8] all fall into P3. These systems require experts to label metadata for each target

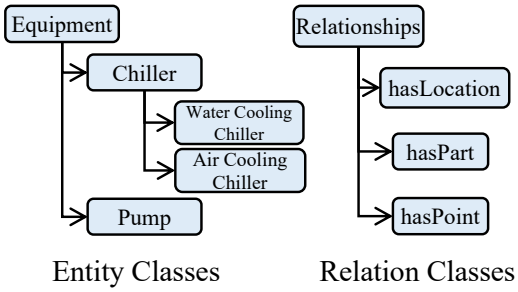


Figure 1: Examples of the Brick Schema

building and this could be demanding in practice. In this paper, we complement existing studies with P1 and P2. We carefully analyze the practical scenarios for P1 and P2, and we develop Cloze, a system that can progressively solve P1 and P2.

Our observation is that this building metadata model generation problem is intrinsically a text analysis problem. We thus leverage the information extraction (IE) paradigm [21]. IE is a type of document processing that can extract structured information from unstructured documents/texts. It has been applied in specific domains, such as clinical notes summarization, and legal claim identification [22, 25]. IE provides us concrete steps to follow as well as perspectives on the challenges and solution approaches to refer to. Specifically, Cloze materializes two main ML tasks of IE, *building entity recognition* [17] and *building relation extraction* [6], and train a building entity recognition model and a building relation extraction model. Cloze overcomes a number of challenges to learn the shared knowledge among source buildings (for P1) and to learn the additional knowledge specific to the target building by its specification file (for P2).

To learn shared knowledge, the challenges are (1) *joint learning*: the metadata of source buildings have joint knowledge, both intra- the two learning tasks on entity recognition and relation extraction, and inter- the two tasks. We develop Bi-LTSM based models to learn the joint knowledge in the words and in the characters of metadata. We also develop multitask learning to learn the joint knowledge between the two tasks; (2) *corpus denoise*: metadata of source buildings often contain information specific to the source buildings. For example, WKGO-CP01 contains the name of the building WKGO. They add noises in learning shared knowledge. We develop a filtering algorithm to detect and denoise these metadata; and (3) *coreference resolution*: a building entity can have multiple metadata references; for example, a chiller can be referred to as *CP*, *CHP*, *CH*, etc. We develop a coreference resolution algorithm to detect the coreference and anaphoric links between text entities.

To learn the additional knowledge of a target building, the challenges are (1) *model transfer*: we first need to preserve the knowledge learned from source buildings. Thus, we develop new Bi-LSTM models and model fine-tuning algorithms to effectively transfer the models of the source buildings into the models of the target building; (2) *additional knowledge learning*: the target building has new entities and also new texts in its own metadata. We leverage the specification file of the target building to learn such knowledge without the need for labeled data; and (3) *disambiguity*: When we preserve the shared knowledge from the source buildings and integrate new knowledge from the target building, ambiguity may

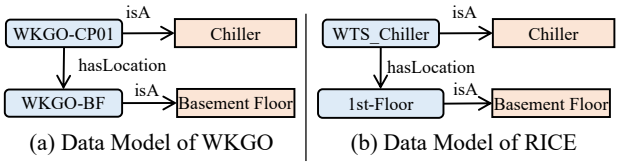


Figure 2: Examples on the building metadata models of WKGO and RICE, both follow the Brick Schema

arise. Again, we leverage the specification file to generate samples to disambiguate the knowledge.

We implement Cloze and integrate it into the Brick eco-system.[1] We evaluate Cloze using six real-world buildings, and two Brick schema versions 1.0 and 1.2. We show that Cloze can serve a set of 11 universal applications in smart buildings (P1) with an accuracy of 96.6% by successfully learning the shared knowledge from source buildings. We show that Cloze can serve arbitrary applications for a target building (P2) with an accuracy of 96.3% by successfully learning the additional knowledge of the target building. Both outperform existing systems, e.g., Scrabble [13].

The contribution of this paper can be summarized as follows:

- We articulate a formulation of the building metadata model generation problem. We analyze three variants that fit different practical scenarios (§2.1).
- We analyze the challenges in the problem, both in learning the shared knowledge of source buildings and in learning the additional knowledge of a target building §2.3. We develop a Cloze system based on the IE paradigm §3 which effectively addresses the challenges.
- We implement Cloze and integrate it into the Brick ecosystem. We evaluate Cloze using the data of six real-world buildings, and two Brick schema versions 1.0 and 1.2. Our results show that Cloze can achieve an accuracy of 96.3%.

2 THE PROBLEM AND CHALLENGES

2.1 The Problem

Before we formally present our problem, we first describe an example on Brick, building metadata model, and the classification of metadata/metadata pairs. Brick defines two types of classes, building entity classes and building relation classes, all are organized in a hierarchy. Figure 1 shows an example. In Brick, the building metadata models of WKGO and RICE are shown in Figure 2. WKGO-CP01 and WTS-Chiller, though with different text conventions, all fall into the Chiller entity class. We need to automatically construct such models and the key steps are to classify metadata into building entity classes and metadata pairs into building relation classes. Figure 3 shows an example that WKGO-BF-CP01 is classified to the entity class *Chiller*; and the relation of WKGO-BF-CP01 and WKGO-BF is classified to the relation class *hasLocation*.

The building metadata model generation problem (P1): given the Brick building metadata schema; the metadata of a set of source buildings and their labels; develop and train (1) a building entity recognition model (BEntity) to classify a piece of metadata

[1]We make our codes available: https://github.com/fangger4396/Cloze

(1) WKGO-BF-CP01, isA, ___A____
A. Chiller B. AHU C. VAV D.Damper

(2) WKGO-BF-CP01, ____C____, WKGO-BF
A. hasPart B.hasPoint **C. hasLocation** D. controls

Figure 3: Examples on building entity recognition and building relation extraction

into an entity class and (2) a building relation extraction (BRelation) model to classify two building metadata into a relation class.

P1 is agnostic to any specific building. It serves the applications that require the metadata to be classified into entity classes that are universal across all buildings. In this paper, we categorize these universal entity classes by the top-level Brick classes in the Brick hierarchy. More specifically, Brick has three root classes Equipment, Location, and Points and we categorize the next two level classes under these three classes as the universal Brick entity classes. In practice, it is sufficient to classify a piece of metadata into its universal Brick entity classes. For example, an application may need the power consumption sensor data of an AHU fan. In building C, AHU_3_Supply_Fan_Power is the power sensor, while in building D, AHU-3_exh_fan_pow is the power sensor. In the regular Brick entity classes, AHU_3_Supply_Fan belongs to the Supply Fan class and the AHU-3_exh_fan belongs to the Exhaust Fan class. However, it is sufficient to categorize AHU_3_Supply_Fan or AHU-3_exh_fan into the Fan class in Brick. Tools such as Mortar [11] and Energon [12] can retrieve the power consumption data correctly following the Fan class (and then its Power Sensor class).

There are also applications that are closely tied with a specific target building. It is then necessary to classify the metadata into regular Brick entity classes. We need additional inputs to learn the knowledge specific to the target building.

One such additional input is the *building metadata specification file*. Figure 4 shows an example.[2] Intuitively, a specification file describes the abbreviation convention of the building metadata.

Note that it is difficult to directly convert the specification file into Brick classes since the specification files of different buildings vary greatly; many descriptions are arbitrarily written and the description in the specification file can even mismatch Brick classes. Nonetheless, specification files can provide information related to the target building. We will use specification files to generate composed metadata of the target building, which can be used to assist model training. We have problem P2:

Problem P2: Problem P1 with an additional input of a metadata specification file of the target building.

We can also pre-label a partial set of the metadata of the target building. This leads to problem P3:

Problem P3: Problem P1 with additional inputs of a set of pre-labeled metadata of the target building.

Existing systems, e.g., Scrabble and ProgSyn, all fall into P3. In these systems, the pre-labeled metadata is critical to achieving good performance. More specifically, these systems adopt active learning methods which allow experts to iteratively label the metadata of the target building. In an iteration, active learning will evaluate

[2]We put two real-world metadata specification files in https://github.com/fangger4396/Specification-File/blob/main/metadata_specification_file.md.

Equipment Code

Equipment Description	Equipment Code
Water Cooling Chiller	CH
Condensing Water Pump	CWP
Condensing Water Valve	CWVLV
…	…

Room Code

Room Description	Room Code
Plant Room	PLANT
Pump Room	PRM
…	…

Function Code

Function Description	Function Code
Chilled Water Flow Rate	CHWFWR
Chilled Water Inlet Temperature	CHWIT
…	…

Figure 4: An example metadata specification file

the expert labels in the current iteration and inform the experts to label the more representative metadata (those with a greater difference from existing labeled metadata) in the next iteration. These approaches, though tried to reduce the amount of pre-labeling effort to a certain extent, intrinsically rely on pre-labeled metadata. As said, seeking expert engineers to label metadata for each target building may be difficult in practice. In this paper, we complement existing studies with P1 and P2. We need to develop new approaches since the solutions used in P3 cannot perform well in P1 and P2.

2.2 Potential Approach: Information Extraction

Without expert labels on metadata, we take an approach to investigate the characteristics of building metadata and develop algorithms to process it. We observe that building metadata processing is intrinsically a text analysis problem. We thus leverage the information extraction (IE) paradigm [21], which can provide us a systematic solution framework. IE is a task of automatically extracting structured information from unstructured documents/texts and it has been widely used in such domains as clinical notes summarization, legal claim identification [22, 25], etc.

Within the IE paradigm, two typical tasks are *name entity recognition* and *relation extraction*. For example, for the sentence "Michelle Obama is very supportive of her husband, Barack Obama", the words "Michelle Obama" and "Barack Obama" are detected as named entities and classified into a public figure class, and the relation between "Barack Obama" and "Michelle Obama" is classified into a spouse class. We argue that IE nicely fits our problem. IE provides concrete steps to follow on building entity recognition and building relation extraction. More importantly, it provides an organized perspective on the challenges and solutions to refer to, e.g., corpus denoise, coreference resolution, model transfer, etc., as we discuss next.

2.3 Challenges

There is no current system for P1 and P2. To show design challenges, we develop Cloze-Infant, a system that directly follows the IE tasks on building entity recognition and building relation extraction.

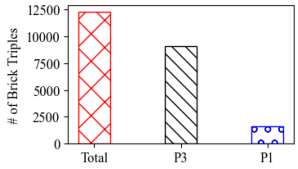


Figure 5: The number of correctly constructed Brick triples for building A

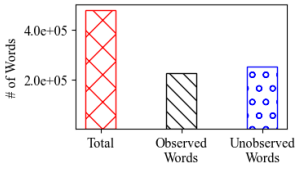


Figure 6: The number of observed and unobserved words in building B

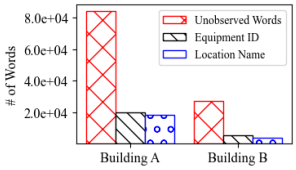


Figure 7: The constituents of unobserved words in building A and B

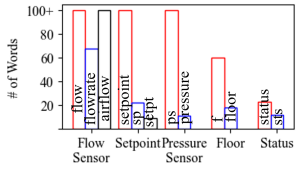


Figure 8: The words refer to the same entity class in building A and B

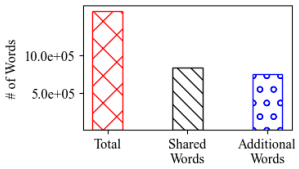


Figure 9: The number of shared words and additional words in building A

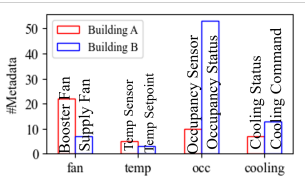


Figure 10: Ambiguity words represents different entities in building A and B

Cloze-Infant: We develop a first system with a building entity recognition (BEntity) model and a building relation extraction (BRelation) model, both based on the Bag of Word (BoW) model [23] commonly used for text analysis.

We evaluate Cloze-Infant in two buildings. Building *A* has 12,260 Brick triples of 135 different classes and building *B* has 4,535 triples of 59 classes. The target building is *A*. Our accuracy metric is: for each Brick triple, it is correctly constructed if all (entity, relation, entity) are consistent with the ground truth.

Figure 5 shows an overall result. Of the 12,260 triples, Cloze-Infant can correctly classify 9,086 if its ML models are trained using the metadata of both the source building and the target building (P3), i.e., the accuracy of 74.1%. Yet if they are trained with source building only (P1), the accuracy is only 13.5%. As a matter of fact, for the systems designed for P3, since they can have labeled data of the target building, their designs seek the assistance of labeled data. For example, they label the most informative metadata of the target building and this improves the performance.

We now investigate the detailed challenges for P1 and P2. Figure 6 shows that 52.8% of the words in building *B* are not observed in building *A*, i.e., the metadata distribution of *A* and *B* differs. This leads to low learning accuracy when there is no labeled data of the target building *A*. Fortunately, we also note that building entities and relations are intrinsically the same; only the "text" representation differs, e.g., *CP*, *CHP* all refer to chillers even though *CP* is used in Building *A* and *CHP* is used in Building *B*. We analyze the metadata of six buildings. We see that there can be shared knowledge, e.g., a pool of text segments refer to a chiller entity, and such knowledge can be learned.

We observe three challenges to learning shared knowledge. We still use the results of Building *A* and *B* for simplicity in presentation.

2.3.1 *Challenges to learn shared knowledge:*

Challenge 1.1 Joint knowledge: There are two learning tasks, building entity recognition and building relation extraction. There is joint knowledge both at the intra-task level and inter-task level. For intra-tasks, metadata needs to be processed in each learning task, and there is joint knowledge among the words and among the characters. For example, "CH", "CP", and "CHP" can represent a chiller at the word level; and they have similar constituents at the character level. For inter-tasks, there is joint knowledge between the building entity recognition and the building relation extraction. For example, if we know that WKGO-BF-CP01 is a chiller, we can predict its associated relation classes with higher accuracy.

Challenge 1.2 Corpus noises: Building metadata often contains building-specific information, i.e., each building can have its own specific information. From the viewpoint of shared knowledge, these become "noise". Figure 7 shows that for the unobserved data in Figure 6, 23.4% of the data are the name of a location, e.g., WKGO: this is specific for building *B*; and 22.0% of the data are equipment ID, e.g., RM3001: this is also specific for building *B*. In total, there are 45.4% of observed data is not knowledge worth sharing.

Challenge 1.3 Entity coreference: A building entity can have diverse metadata references; for example, a chiller can be referred to as *CP*, *CHP*, *CH*, *Chiller*, etc. A basic BoW model cannot easily classify such metadata into the chiller entity class. Figure 8 show a few coreference examples of building *A* and *B*. Note that there are even three or four coreferences of the same entity in two buildings, e.g., flow sensor has been referred to as *flow*, *flowrate*, and *airflow*. As can be imaged, the amount of coreference increases significantly when the number of buildings increases.

To learn the additional knowledge of a target building, we leverage the specification files that widely exist in buildings. They have information of a target building. We observe three more challenges.

2.3.2 *Challenges to learn additional knowledge of a target building:*

Challenge 2.1 Learned knowledge transfer: We need to learn the building entity recognition model and the building relation extraction model for the target building. In addition to integrating new knowledge of the target building, we also need to transfer the knowledge learned from source buildings.

Challenge 2.2 Learning additional knowledge: A target building has specific entities and specific texts. These are not noise, since we need to serve targeted applications to this building. On the contrary, they provide useful knowledge. Figure 9 shows that 48.2% of words in building *A* can be additional to building *B*. It is difficult to learn such additional knowledge of the target building directly due to the lack of labeled metadata from the target building.

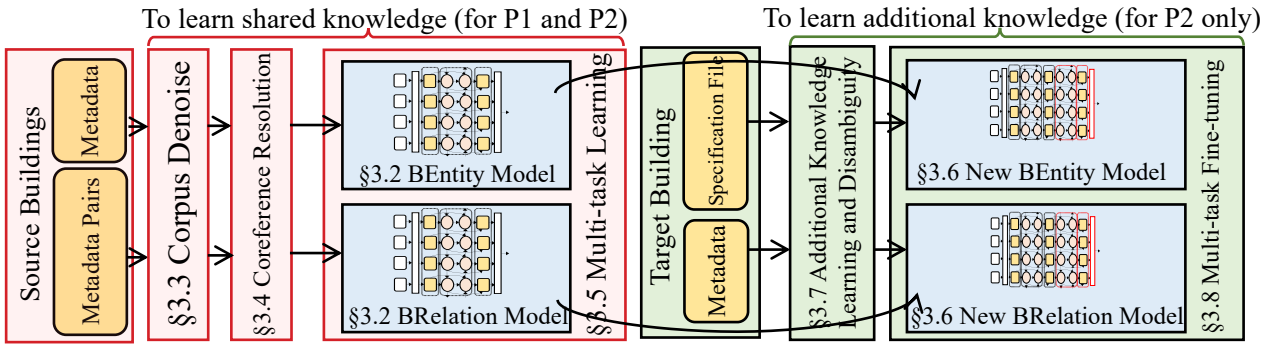


Figure 11: The Cloze system

With the metadata specification file, how the additional knowledge should be learned is yet to be developed.

Challenge 2.3 Ambiguity in the learned knowledge: When we preserve the shared knowledge from the source buildings and integrate new knowledge from the target building, ambiguity arises. For example, in the target building *A*, there can be a new chiller entity class, i.e., a water cooling chiller entity class and text *CH* in *A* refers to a water cooling chiller. In the learned knowledge, *CH* refers to a chiller entity class. The knowledge that *CH* is a chiller can benefit the classification of *CH* into the water cooling chiller, yet ambiguity should be solved finally. Figure 10 shows many ambiguity examples of building *A* and *B*.

In summary, there are many challenges in IE. This section analyzes the existence of the specific challenges we face in building metadata through specific examples and quantitative measurement.

3 THE CLOZE SYSTEM

3.1 System Overview

Figure 11 shows the modular design of the Cloze system. Cloze first tokenizes the metadata into word tokens (not shown). Cloze has the BEntity and BRelation models and the modules to solve the challenges. Cloze can solve both P1 and P2.

The core of Cloze is to train two models (blue): the BEntity model and the BRelation model. We develop the two models based on the Bi-LSTM model, a text analysis neural network model widely used in machine translation and sentiment analysis [33]. Other text analysis models include Conditional Random Fields (CRF) [20] and Hidden Markov Models (HMM) [16]. CRF is suitable for character-level text analysis and HMM is suitable for scenarios where adjacent texts have dependencies. We choose the Bi-LSTM since: (1) building metadata are meaningful both in the forward order and in the backward order. For example, both "WCC-CP-01" and "01-CP-WCC" can be used to represent a chiller; and (2) we need joint training of the entity recognition task and relation extraction task. Therefore, we need a model that can perform well in both tasks. Bi-LSTM fits both (1) and (2). We develop the detailed structures of the BEntity and BRelation models in §3.2.

Cloze has three modules (red) addressing the three challenges to learning the shared knowledge of source buildings (P1): (1) a corpus denoise module for removing the noisy words that hamper learning (§3.3), (2) a coreference resolution module for detecting the metadata that refers to the same entity class (§3.4), and (3) a multi-task learning module to joint train the BEntity and BRelation models (§3.5).

Cloze has three modules (green) addressing the three challenges for transferring the shared knowledge of the source buildings and learning the additional knowledge of the target building: (1) new BEntity and BRelation models designed to transfer the shared knowledge of the source building §3.6; (2) an additional knowledge learning and disambiguity module to learn new knowledge and discharge ambiguity using the specification file (§3.7); and (3) a multi-task fine-tuning module to integrate new knowledge (§3.8).

Cloze in Execution: The execution process of Cloze follows a regular ML process. There is a training phase and an inference phase. In the training phase of P1, the Cloze system trains the BEntity model and the BRelation model by the metadata from source buildings. In the training phase of P2, the Cloze system trains the new BEntity model and the new BRelation model by the metadata from source buildings as well as the composed metadata of the target building that are generated from the specification file. In the inference phase, the BEntity and BRelation models are used to infer the entity class of a piece of metadata and the relation class of a metadata pair.

3.2 The BEntity and BRelation Model Structure

The BEntity and BRelation models are based on the Bi-LSTM model. The special characteristic of our text analysis is that we need to recognize the words both on a word level and on a character level. Specifically, "CH", "CP", and "CHP" can represent a chiller at the word level; and they have similar constituents at the character level (see Challenge 1.1).

We develop our models to have the capability both to recognize words and to capture the character features of how the words are constructed by characters. The model structure of the BEntity is shown in Figure 12. First, the character tokens of a word are transformed into character embeddings. The Character Bi-LSTM layer takes the character embeddings as inputs and outputs the character representation of a word. Second, the character representation and the word embeddings are concatenated. The word Bi-LSTM layer takes the concatenation as inputs and outputs a word representation. Third, dense layers with sigmoid activation functions are added to capture the non-linearity of the features. Fourth, we use the softmax function as the activation function for the output layer. It outputs the Brick entity class of the input metadata.

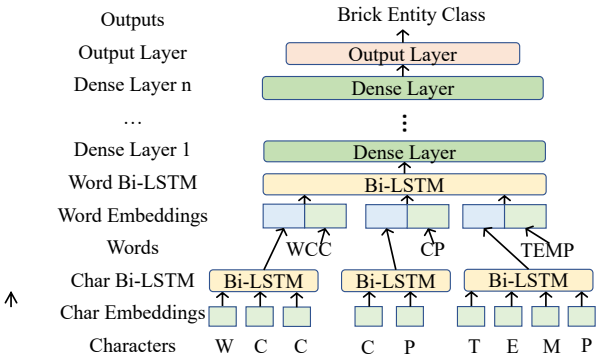


Figure 12: The neural network structure of BEntity

The design of the BRelation model is similar to that of the BEntity model. The difference is that the inputs of the BRelation model are metadata pairs. As such, we design the character embedding layer and the word embedding layer to be capable to embed metadata pairs. We use the same word embedding layer and the character embedding layer for both metadata in the pairs; this is effective in learning better representations.

3.3 Corpus Denoise

Each source building can have its own specific information. From the viewpoint of shared knowledge, these are "noise". We choose a denoise approach in text analysis [19] where we detect infrequent words in the metadata as "noise" and remove them.

Intrinsically, we need to define an appropriate frequency of the words so as to remove the "noise" while maintaining the shared knowledge. We observe two types of noisy words for building metadata: (1) the words that have low frequency in each source building. For example, "Room3001" appears in a number of buildings. Though its total number adds up, it only appears zero or very few times in each building; and (2) the words that have a high frequency in one source building but have a low overall frequency. For example, "NW" is significant in the metadata of building "WKGO", since it represents the north wing of the building. However, it never appears in other source buildings. We choose to use the term-frequency (TF) score [5] to estimate the former and term-frequency-inverse-document-frequency (TF-IDF) score [4] to estimate the latter.

The TF score of a word w, TF_w is:

$$TF_w = \frac{\sum_{b \in B} f_{w,b}}{\sum_{b \in B} \sum_{w' \in b} f_{w',b}} \tag{1}$$

Here, B is the set of buildings and $b \in B$. $f_{w,b}$ is the word-counts of w in b. Intuitively, TF_w shows the total frequency of w in all words.

The TF-IDF score of a word w in building b, $TFIDF_{w,b}$ is:

$$TFIDF_{w,b} = \frac{f_{w,b}}{\sum_{w' \in b} f_{w',b}} \times log(\frac{|B|}{B_w}) \tag{2}$$

Here, $|B|$ is the number of total buildings, B_w is the number of buildings that contains the word w. Intuitively, $TFIDF_{w,b}$ shows the frequency of w in a specific building, with a logarithmic factor.

In our implementation, we follow the procedure in [19] to first calculate the TF score and TF-IDF scores; and then detect and remove the words according to pre-defined thresholds.

3.4 Coreference Resolution

Coreference resolution is heavily studied in NLP. There are advanced algorithms such as the Hobbs algorithm [15] and the centering theory algorithm [29] that apply complex syntax and linguistic analysis. Our scenario is much simpler. We develop a coreference resolution algorithm where we first detect the words in the building metadata that refer to the same entity class by a clustering-based algorithm and then replace the words with their coreference.

Our clustering-based algorithm draws upon [9], where we calculate the distance between words and cluster the words with small distances. More specifically, we choose the edit distance between two words as the metric of distance, and we iteratively merge the cluster of words where the edit distances of the pairs are smaller than a certain threshold, similar to the algorithm in [9].

We then replace the words referring to the same entity with their coreference so that all these representations can be learned by the BEntity model. We choose a data augmentation method to construct new building metadata to replace a word in the metadata with one of its coreference words. For example, for the metadata "BF-CP01" (labeled as a chiller), we replace "CP" with "CH" and generate a new metadata "BF-CH01'" (also labeled as a chiller). By training with the augmented labeled metadata, the BEntity model can recognize that both "CH" and "CP" refer to a chiller.

3.5 Multi-task Learning

There is joint knowledge between the building entity recognition learning task and the building relation extraction learning task, i.e., if the entity class of a piece of metadata is known, it helps to learn the relation class, and vice versa.

We use multi-task learning to learn the joint knowledge between tasks. We choose to share the parameters of the BEntity and BRelation models. Specifically, we share the word embedding layer and character embedding layer since these two layers of the two models have the same structure. During the training process of the two models, we copy the parameters of the word embedding layer and character embedding layer of one model to the corresponding layers of the other model in each iteration.

3.6 New BEntity and BRelation Model Structure

For the problem to serve a target building, we develop a new BEntity and a new BRelation model. The new models need to preserve the knowledge learned from source buildings and can be used to learn new knowledge of a target building. We develop the new models by extending the BEntity and BRelation models with additional neural network layers so that the knowledge in the original models can be transferred to the new models.

In Figure 13, we show the new BEntity model. We preserve all the layers and the trained parameters of the BEntity model except that we remove the output layer. We add an additional Bi-LSTM after the BEntity model to learn the additional forward and backward information from the new metadata of a target building. We reshape the output of the last dense layer of the original BEntity model to convert the output into a sequence. Thus, this sequence can be used as the input of the additional Bi-LSTM layer. We add a few more dense layers to capture the non-linearity of the features. Finally, we again use the softmax function as the activation function for the

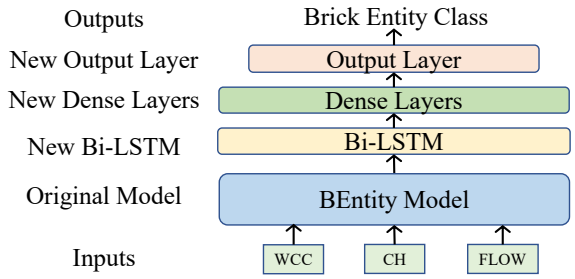


Figure 13: The structure of the new BEntity model

output layer. The design of the extended layers of the new BRelation model is the same as that of the new BEntity model.

3.7 Additional Knowledge Learning and Disambiguity

The target building can have additional knowledge that does not exist in the knowledge of the source buildings. For example, *CWP* represents a condensing water pump, i.e., it falls into a Water Pump entity class in Brick, yet it never appears in any of the source buildings (in source buildings, *Pump*, *PP*, etc. have been learned). The target building can also introduce ambiguity. For example, *CH* represents a water cooling chiller in this target building; yet *CH* represents a chiller in source buildings.

To learn the additional knowledge from the target building, we seek assistance from the metadata specification files. As said, however, it is difficult to directly convert the specification file into Brick classes since the specification files of different buildings vary greatly; many descriptions are arbitrarily written and the descriptions in the specification files can mismatch Brick classes. For example, the description of metadata *CHWIT* can be "Chilled Water Inlet Temperature", whereas its correct Brick class should be Chilled Water Supply Temperature Sensor class.

Nevertheless, the words in the descriptions of the specification files can expose useful information in the metadata of the target building. We thus use the specification file to compose labeled metadata of the target building. This composed metadata with labels can then be used in model training. There are two steps.

First, we classify the metadata into one of the three categories under the root classes, Location, Equipment, and Points. A piece of metadata can contain the words of multiple root classes. For example, metadata BF-CP-TEMP contains Location (Basement Floor), Equipment (Chiller), and Points (Temperature Sensor). We develop priorities for these three root classes. From a high priority to a low priority, they are the Point entity classes, the Equipment entity classes, and the Location entity classes. We believe that these priorities intrinsically reflect the common convention used in metadata development. For example, metadata BF-CP-TEMP should be classified into one of the Brick Point entity classes, instead of one of the Equipment entity classes or one of the Location entity classes. We develop a standard text-matching algorithm to match the description of the metadata and one category of Brick entity classes by traversing the Brick entity classes in the priority of the Point, Equipment, and Location entity classes.

Second, with the category determined, we compose the labels for the metadata, i.e., to label the Brick entity class. This is a maximum matching problem in text mining. More specifically, for certain metadata, we extract the set of words in the description of this metadata and the set of words in the Brick entity classes where we maximize the overlap between the two sets of words.

3.8 Multi-task Fine-tuning

Finally, we train (fine-tune) the new BEntity and BRelation models by using the newly labeled metadata of the target building. These labeled metadata provide the knowledge which can be learned into the new BEntity and BRelation models. We still apply the multi-task learning of the two models.

4 EVALUATION

In this section, we evaluate Cloze with real-world buildings. We first present the methodology used for the evaluation and we then present the evaluation results for P1, P2, as well as the contribution from each of our design components.

4.1 Methodology

Datasets: We evaluate Cloze with six real-world buildings. Four are office buildings and two are campus buildings. Their specifications are shown in Table 1. The number of metadata varies from 479 to 6,964 while the number of Brick entity classes varies from 21 to 129. The building metadata are different in the length of words and characters. Five buildings were based on a public dataset [2] and building C is based on a private dataset that we are yet to obtain the authority to publish this dataset.

Metrics: We evaluate the accuracy of the building metadata model, i.e., whether the triple (entity, relation, entity) is correct. This requires that both the entity classification and the relation classification be correct. We evaluate two types of applications: (1) universal applications across buildings (P1). We analyze the literature and choose 11 universal applications, see Table 2. The entity classification is correct if the metadata is correctly classified into the universal Brick entity classes, i.e., the top two level entity classes in the Brick hierarchy. The ground truths of the respective applications are shown in Table 2; and (2) applications for a target building (P2). The entity classification is correct if the metadata is correctly classified into the regular Brick entity classes.

Experiment Setup: For P1, we conducted two sets of experiments. Table 3 shows the experiment set up on the training datasets and the testing datasets. We chose Building C and D for testing since their metadata cover all 11 applications, yet other buildings each miss some different applications. To simplify the presentation of our results, we only conduct the testing on Building C and D. For P2, we conduct six sets of experiments. Table 4 shows the experiment set up on the training datasets and the testing datasets.

For P1, we conduct data preprocessing on the training datasets according to §3.3 (corpus denoise) and §3.4 (coreference resolution). For P2, we conduct data preprocessing on the training datasets according to §3.3, §3.4, and §3.7 where we use the specification file to generate composed metadata.

Metadata Schema We evaluate Cloze using two Brick schema versions, 1.0 and 1.2. The Brick schema 1.0 is an early Brick version

Table 1: The building metadata datasets from six buildings

Building	Property of Metadata			Property of Brick Class		Ground Truth	
	Num. of Metadata	Num. of Metadata Pairs	Ave. Words	Num. of Entity Class	Num. of Relation Class	Num. of Type-1	Num. of Type-2
A	6964	36485	13.7	56	11	-	43349
B	3868	8392	10.8	129	6	-	12260
C	1796	2739	11.9	49	10	645	4535
D	1410	2078	8.5	27	11	975	3488
E	690	5235	14.4	21	13	-	5925
F	479	718	6.9	33	10	-	1197

Table 2: Universal Building Applications

ID	Building Applications	Ground Truth
(1)	Chiller Profiling [35]	Chiller, Water Temperature Sensor, Flow Sensor, Power Sensor, (and their subclasses)
(2)	FDD for Chillers [34]	
(3)	Load Forecasting [30]	
(4)	Energy Inefficiency Detection [27]	
(5)	Indoor Air Quality [32]	Zone, Room, CO2 Sensor, Temperature Sensor, Flow Sensor, (and their subclasses)
(6)	Building Integrated Control [24]	
(7)	Ventilation Prediction [28]	
(8)	FDD for AHU [18]	AHU, VAV, Temperature Sensor, Flow Sensor, Power Sensor, (and their subclasses)
(9)	FDD for VAV [31]	
(10)	Energy Consumption Prediction [3]	
(11)	Model Predictive Control [14]	

released in 2017. As presented, Brick defines a set of entity classes and a set of relation classes. In Brick 1.0, 2025 entity classes were defined, including chiller, AHU, VAV, etc; and 12 relation classes were defined, including hasPart, hasPoint, etc. It has been used in a number of buildings, e.g., RICE, GHC, etc. The current version of the Brick schema is Brick 1.2, which was released in 2021. As compared to Brick 1.0, Brick 1.2 removed some classes redundantly defined in Brick 1.0. For example, Brick 1.0 defined a Current Cooling Setpoint class, an AHU class, and an AHU Current Cooling Setpoint class. Brick 1.2 removed AHU Current Cooling Setpoint because AHU Current Cooling Setpoint can be specified as a Current Cooling Setpoint class of the AHU class. In Brick 1.2, the total number of entity classes and relation classes is 1034 and 24 respectively. The default version we use is Brick 1.2.

Baselines We implement Cloze and compare it to Scrabble [13], a state-of-the-art building metadata model generation system. Scrabble has two stages. First, it infers the Brick entity class of a piece of metadata. This is done by an ML model developed based on conditional random fields. Second, it infers the Brick relation class of a pair of metadata by using a set of predefined rules. As said, Scrabble falls into problem P3, i.e., its ML model is trained by the metadata of source buildings and then retrained by a subset of the metadata of the target building labeled by Brick experts. For a fair comparison, we also implement Scrabble in a P1 mode with only labeled data for source buildings and a P2 mode where we manually label a fraction of metadata randomly selected from the target building. We evaluated both 10% and 20% of labeled metadata, denoted as Scrabble-10% and Scrabble-20%. The default setting for Scrabble in our paper is 20%. We also show the performance of Cloze-infant for benchmarking.

4.2 Evaluation for Universal Applications (P1)

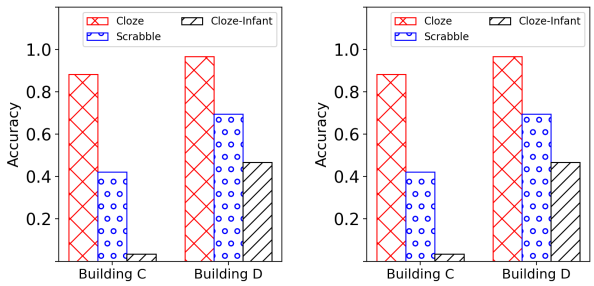
Figure 14 (a) shows the performance of Cloze, Scrabble, and Cloze-infant. We can see that the accuracy of Scrabble and Cloze-infant

Table 3: Experiment setup for P1

Experiment #	Training Datasets	Testing Dataset
1	Building A,B,D,E,F	Building C
2	Building A,B,C,E,F	Building D

Table 4: Experiment setup for P2

Experiment #	Training Datasets	Testing Dataset
1	Building B,C,D,E,F	Building A
2	Building A,C,D,E,F	Building B
3	Building A,B,D,E,F	Building C
4	Building A,B,C,E,F	Building D
5	Building A,B,C,D,F	Building E
6	Building A,B,C,D,E	Building F



(a) Brick Schema 1.2 (b) Brick Schema 1.0

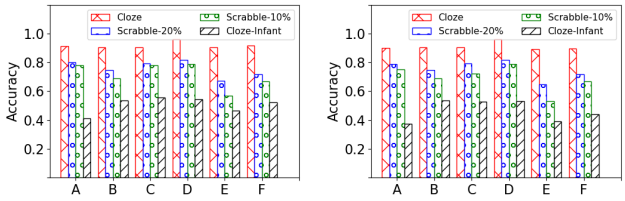
Figure 14: Comparison of Cloze, Scrabble, Cloze-infant in P1

is much lower than Cloze. The accuracy of Cloze on Building *C* is 90.54% and Building *D* is 96.62% As a comparison, the accuracy of Scrabble and Cloze-infant on Building *C* is 42.19% and 3.4%, respectively; and on Building *D* is 69.55% and 46.77%, respectively. This clearly shows that Cloze has the capability to effectively learn the shared knowledge of source buildings.

Figure 14 (b) compares the performance of Cloze, Scrabble, and Cloze-infant under Brick schema 1.0. We see that the accuracy of all three systems is the same as the performance under Brick schema 1.2. This reflects Cloze can serve universal applications where only general entity classes are required.

4.3 Evaluation for Regular Applications for a Target Building (P2)

Figure 15 (a) shows the accuracy of Cloze, Scrabble, and Cloze-infant for each building. We can see that Cloze outperforms Scrabble and Cloze-infant on all six buildings. The accuracy of Cloze for each of the six buildings are 91.14%, 90.68%, 90.32%, 96.29%, 90.35%, 91.93%. As a comparison, the accuracy of both Scrabble and Cloze-infant are much lower. For example, the accuracy of Scrabble-10%, Scrabble-20% and Cloze-infant is 80.07%, 78.17% and 41.15% for building *A* respectively. This clearly shows that Cloze is successful



(a) Brick Schema 1.2 (b) Brick Schema 1.0

Figure 15: Comparison of Cloze, Scrabble, Cloze-infant in P2

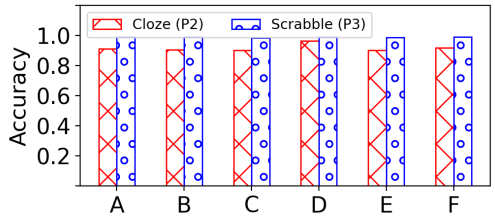


Figure 16: Comparison of Cloze (P2) and Scrabble (P3)

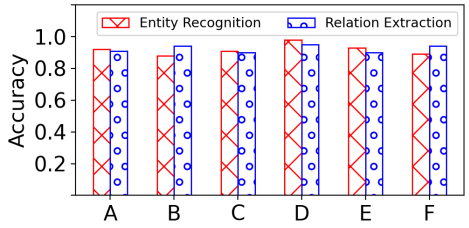


Figure 17: Accuracy of the entity recognition task and relation extraction task of Cloze

in transferring the shared knowledge of the source buildings and learning new knowledge in the target building.

Figure 15 (b) compares the accuracy of Cloze, Scrabble, and Cloze-infant under Brick schema 1.0. We can see that Cloze again has a good performance. More specifically, the accuracy of Cloze is 90.05%, 90.66%, 90.26%, 96.17%, 89.16%, 89.70%. Cloze outperforms Scrabble and Cloze-infant, e.g., in Building *A*, the accuracy of Scrabble-10%, Scrabble-20%, and Cloze-infant, is only 78.69%, 75.43% and 37.45%. We see that the performance is slightly lower than that of Brick schema 1.2. Recall that Brick schema 1.2 has a more concise list of entity classes. Intuitively, this makes entity recognition and relation extraction easier. The accuracy of Scrabble and Cloze-infant is less than Cloze and also decreases.

We now compare Cloze directly with Scrabble where Scrabble is trained with expert labeled metadata for the target building, i.e., P3. More specifically, Scrabble labels the most informative metadata of the target building. We follow the method developed in Scrabble [13]. This method can select the metadata that can lead to the largest reduction in the accuracy loss. Domain expertise can then label these metadata. Figure 16 shows that the performance of Scrabble increases greatly for all six buildings. For example, for building *A*, the accuracy can be as high as 99.19%. This shows that Scrabble

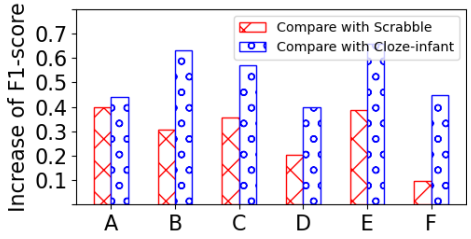


Figure 18: Comparison of Cloze, Scrabble and Cloze-infant under Macro-F1 score

indeed successfully selects the appropriate metadata of the target building for domain experts to label; and then learns the knowledge. In practice, if domain expertise in building metadata schema can be sought, Scrabble provides a good performance. We also see that the performance of Cloze is relatively comparable to Scrabble, with an accuracy of 91.14% for building *A*. Cloze achieves such an accuracy with a set of designs in information extraction of building metadata; which substitutes for expert labeling.

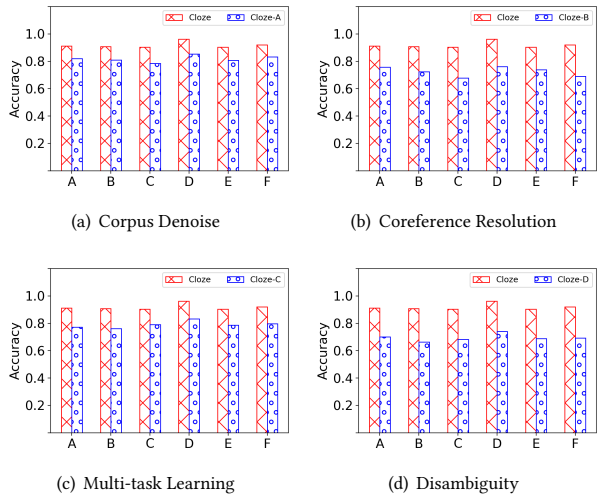
We also compare Cloze, Scrabble, and Cloze-infant under the Marco-F1 score [26]. Figure 18 shows that Cloze outperforms Scrabble and Cloze-infant over all six buildings. As an example, in Building B, Cloze outperforms Scrabble and Cloze-infant for 30.86% and 63.47% respectively.

4.4 Component Analysis

We now study our designs on each individual component of our system. We first study the performance of entity recognition (the BEntity model) and relation extraction (the BRelation model). We then study the performance of the four modules of Cloze to understand their contributions to the performance of the system. For simplicity, our evaluation is based on P2, which has all modules. More specifically, we develop **Cloze-A:** Cloze without corpus denoise; **Cloze-B:** Cloze without coreference resolution; **Cloze-C:** Cloze without multi-task learning, i.e., we learn the BEntity and BRelation models independently; and **Cloze-D:** Cloze without disambiguity. Note that the other two modules, i.e., the additional knowledge learning and fine-tuning in §3 are compulsory.

Figure 17 shows the accuracy of the entity recognition task and the relation extraction task in Cloze. We see that Cloze has a high accuracy for both tasks in all six buildings. For example, in building *A*, the entity recognition task achieves an accuracy of 92.27% and the relation extraction task achieves an accuracy of 90.96%. These illustrate our designs for both tasks are successful.

Figure 19 shows the contributions of each component to the performance of Cloze in six buildings. Figure 19 (a) shows the performance of Cloze without corpus denoise (i.e., Cloze-A). We see that the performance of Cloze decreases by 9%-12%. For example, the accuracy of Cloze-A for building *A* is 82.12% while the accuracy of Cloze is 91.14%. In other words, the contribution of corpus denoise is 9.02% for building *A*. Figure 19 (b) shows the performance of Cloze without coreference resolution. The performance of Cloze decreases by 16%-23%. For example, for building *A*, the accuracy of Cloze-B is 75.78% which reveals that coreference resolution can increase the accuracy by 15.36%. Figure 19 (c) shows the performance of Cloze without multi-task learning. The performance of Cloze



(a) Corpus Denoise (b) Coreference Resolution

(c) Multi-task Learning (d) Disambiguity

Figure 19: Component analysis for the individual modules of Cloze

decreases by 11%-14%. For example, in building *A*, the accuracy of Cloze-C is 77.28%; this reflects that multi-task learning can improve the accuracy by 13.86%. Figure 19 (d) shows the performance of Cloze without disambuiguity. We see that the performance of Cloze decreases by 21%-25%. For example, for building *A*, the accuracy of Cloze-D is 69.67%. It reflects that the contribution of disambiguity is 21.47%. These show that each of our components is necessary and our designs successfully improve the performance of our system.

5 CONCLUSION

In this paper, we developed Cloze, a system that can convert the metadata of a building in an ad hoc convention into a building metadata model that follows a standard building data schema, e.g., Brick. We show that the problem is intrinsically an information extraction problem. Following the information extraction paradigm, we analyzed with quantitative measurement a set of challenges that building metadata specifically faces, e.g., corpus denoise, coreference resolution, ambiguity, etc. We developed a system Cloze with a set of algorithm solutions. We evaluate Cloze using data from six real-world buildings. Our evaluation showed that our designs were successful in addressing the challenges.

ACKNOWLEDGEMENT

The authors are indebted to the anonymous reviewers and shepherd for their constructive comments; and their time and efforts guiding this paper into a better shape. This work is supported by RGC GRF 15210119, 15209220, 15200321, 15201322, ITF ITSP-ITS/070/19FP, C5018-20G of Hong Kong.

REFERENCES

[1] 2022. Project Haystack. https://project-haystack.org/.
[2] 2022. Public Brick Datasets. https://brickschema.org/resources.
[3] A. Afram, F. Janabi-Sharifi, et al. 2017. Artificial neural network (ANN) based model predictive control (MPC) and optimization of HVAC systems: A state of the art review and case study of a residential HVAC system. *Energy and Buildings* (2017).
[4] A. Aizawa. 2003. An information-theoretic perspective of tf–idf measures. *Information Processing & Management* (2003).
[5] N. Azam and J. Yao. 2012. Comparison of term frequency and document frequency based feature selection metrics in text categorization. *Expert Systems with Applications* (2012).
[6] N. Bach and S. Badaskar. 2007. A review of relation extraction. *Literature review for Language and Statistics II* (2007).
[7] B. Balaji, A. Bhattacharya, et al. 2016. Brick: Towards a Unified Metadata Schema For Buildings. In *Proc. ACM BuildSys'16.*
[8] A. Bhattacharya, D. Hong, et al. 2015. Automated metadata construction to support portable building applications. In *Proc. ACM BuildSys'15.*
[9] C. Cardie and K. Wagstaff. 1999. Noun phrase coreference as clustering. In *1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora.*
[10] A. Daissaoui, A. Boulmakoul, L. Karim, and A. Lbath. 2020. IoT and big data analytics for smart buildings: a survey. *Procedia computer science* (2020).
[11] G. Fierro, M. Pritoni, et al. 2018. Mortar: An Open Testbed for Portable Building Analytics *(BuildSys '18).*
[12] F. He, Y. Deng, Y. Xu, C. Xu, D. Hong, and D. Wang. 2021. Energon: A Data Acquisition System for Portable Building Analytics. In *Proc. ACM e-Energy' 21.*
[13] J. Koh, D. Sengupta, et al. 2017. Scrabble: converting unstructured metadata into brick for many buildings. In *Proc. ACM BuildSys'17.* 1–2.
[14] A. Kusiak, G. Xu, and Z. Zhang. 2014. Minimization of energy consumption in HVAC systems with data-driven models and an interior-point method. *Energy Conversion and Management* 85 (2014), 146–153.
[15] S. Lappin and H. J. Leass. 1994. An algorithm for pronominal anaphora resolution. *Computational linguistics* (1994).
[16] S. Morwal, N. Jahan, and D. Chopra. 2012. Named entity recognition using hidden Markov model (HMM). *International Journal on Natural Language Computing (IJNLC) Vol* (2012).
[17] D. Nadeau and S. Sekine. 2007. A survey of named entity recognition and classification. *Lingvisticae Investigationes* (2007).
[18] M. Najafi, D. M. Auslander, P. L. Bartlett, P. Haves, and M. D. Sohn. 2012. Application of machine learning in the fault diagnostics of air handling units. *Applied Energy* 96 (2012), 347–358.
[19] L. H. Patil and M. Atique. 2013. A novel approach for feature selection method TF-IDF in document clustering. In *2013 3rd IEEE international advance computing conference (IACC).* IEEE.
[20] N. Patil, A. Patil, and B. V. Pawar. 2020. Named entity recognition using conditional random fields. *Procedia Computer Science* (2020).
[21] J. Piskorski and R. Yangarber. 2013. Information extraction: Past, present and future. In *Multi-source, multilingual information extraction and summarization.*
[22] W. Ruan, N. Appasani, et al. 2018. Pictorial visualization of EMR summary interface and medical information extraction of clinical notes. In *2018 IEEE international conference on computational intelligence and virtual environments for measurement systems and applications (CIVEMSA).*
[23] A. Sethy and B. Ramabhadran. 2008. Bag-of-word normalized n-gram models. In *Ninth Annual Conference of the International Speech Communication Association.*
[24] E. Shen, J. Hu, and M. Patel. 2014. Energy and visual comfort analysis of lighting and daylight control strategies. *Building and Environment* 78 (2014), 155–170.
[25] M. Surdeanu, R. Nallapati, and C. Manning. 2010. Legal claim identification: Information extraction with hierarchically labeled data. In *Workshop Programme.*
[26] K. Takahashi, K. Yamamoto, A. Kuchiba, and T. Koyama. 2022. Confidence interval for micro-averaged F1 and macro-averaged F1 scores. *Applied Intelligence* (2022).
[27] H. Talei, D. Benhaddou, et al. 2021. Smart Building Energy Inefficiencies Detection through Time Series Analysis and Unsupervised Machine Learning. *Energies* (2021).
[28] G. Tan and L. R. Glicksman. 2005. Application of integrating multi-zone model with CFD simulation to natural ventilation prediction. *Energy and Buildings* (2005).
[29] J. P. Walker, M. I. Walker, et al. 1998. *Centering theory in discourse.* Oxford University Press.
[30] L. Wang, E. W. M. Lee, R. K. K. Yuen, and W. Feng. 2019. Cooling load forecasting-based predictive optimisation for chiller plants. *Energy and Buildings* (2019).
[31] S. Wang and J. Qin. 2005. Sensor fault detection and validation of VAV terminals in air conditioning systems. *Energy Conversion and Management* (2005).
[32] W. Wei, O. Ramalho, L. Malingre, S. Sivanantham, J. C. Little, and C. Mandin. 2019. Machine learning and statistical models for predicting indoor air quality. *Indoor Air* 29, 5 (2019), 704–726.
[33] G.. Xu, Y. Meng, Xi. Qiu, Z. Yu, and X. Wu. 2019. Sentiment analysis of comment texts based on BiLSTM. *Ieee Access* (2019).
[34] K. Yan, Z. Ji, and W. Shen. 2017. Online fault detection methods for chillers combining extended kalman filter and recursive one-class SVM. *Neurocomputing* 228 (2017), 205–212.
[35] Z. Zheng, Q. Chen, C. Fan, N. Guan, A. Vishwanath, D. Wang, and F. Liu. 2018. Data Driven Chiller Sequencing for Reducing HVAC Electricity Consumption in Commercial Buildings. In *Proc. ACM e-Energy '18.* 236–248.