SeNsER: Learning Cross-Building Sensor Metadata Tagger

Yang Jiao[†], **Jiacheng Li**[†], **Jiaman Wu, Dezhi Hong, Rajesh Gupta, Jingbo Shang**
University of California, San Diego
{yajiao, j9li, j4wu, dehong, gupta, jshang}@eng.ucsd.edu
[†] Equal Contribution

Abstract

Sensor metadata tagging, akin to the named entity recognition task, provides key contextual information (e.g., measurement type and location) about sensors for running smart building applications. Unfortunately, sensor metadata in different buildings often follows distinct naming conventions. Therefore, learning a tagger currently requires extensive annotations on a per building basis. In this work, we propose a novel framework, SeNsER, which learns a sensor metadata tagger for a new building based on its raw metadata and some existing fully annotated building. It leverages the commonality between different buildings: At the character level, it employs bidirectional neural language models to capture the shared underlying patterns between two buildings and thus regularizes the feature learning process; At the word level, it leverages as features the k-mers existing in the fully annotated building. During inference, we further incorporate the information obtained from sources such as Wikipedia as prior knowledge. As a result, SeNsER shows promising results in extensive experiments on multiple real-world buildings.

1 Introduction

Sensor metadata tagging aims at understanding the context (e.g., sensor function and location) of a sensor from its name, which is essential to any smart building technologies (Wang et al., 2018). As illustrated in Figure 1, sensor metadata is typically a concatenation of esoteric abbreviations, each encoding specific information about the sensor, including what they measure/control, where they are located, how they are related to each other, etc. For example, a sensor name `SODA1R430_ART` conveys: the building name (`SOD`), air conditioning equipment ID (`A1`), room ID (`R430`), and the measurement type, which is area room temperature (`ART`). Running any application would require such

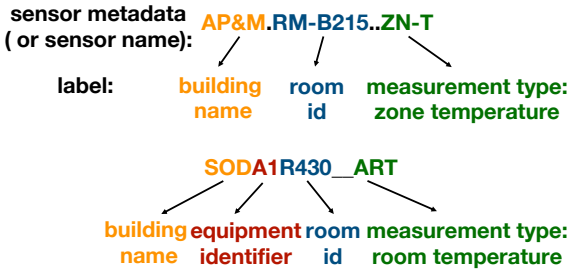

Figure 1: Example sensor metadata from two buildings. Sensor name tagging aims to partition a sensor name into segments (shown in color) that encode key contextual information about sensors. Different buildings adopt distinct vocabularies and naming conventions.

contextual information; for example, to detect overcooled rooms, one needs the temperature and the target temperature set for each room.

Currently, learning a metadata tagger for a new building in practice requires extensive human annotations, thus remaining a bottleneck in deploying smart building techniques widely and efficiently (Wang et al., 2018). This is due to the fact that sensor metadata is curated by building-specific vendors, and that their naming conventions vary drastically across buildings, as shown in Figure 1. Anecdotally, annotating one sensor name may cost several hundred dollars, and it takes weeks to do so for one typical building with thousands of sensing and control points. This manual approach is clearly neither economical nor scalable, and it calls for an automated solution.

As there usually exist buildings that are already tagged, leveraging this information could potentially expedite the tagging process in a new building. Thus, in this paper, we seek to answer the following question: Can we learn a sensor metadata tagger for a new building based on its raw metadata and some existing fully annotated building(s)?

Our problem faces unique challenges, despite its similarity with named entity recognition

(NER) (Tjong Kim Sang and De Meulder, 2003). First, lacking pre-processing tools (e.g., tokenizer) for the building domain, we have only raw character sequences as input to work with (rather than "word" sequences as input), and thus state-of-the-art NER models (Akbik et al., 2018; Peters et al., 2018; Devlin et al., 2018) do not apply. The choice of taggers is therefore confined to only those working at the *character level*. Secondly, the heterogeneity of sensor names in source and target buildings hurts the performance of existing character-level taggers (e.g., Char-LSTM-CRF in Figure 2), resulting in unsatisfactory results. Last, one building typically has "only" a few thousand sensor names, and each sensor name has fewer than two dozen characters; however, there are more than 100 types for tagging.

Recognizing these challenges, we propose a novel framework – SeNsER. At the character level, together with the tagging objective function on the source building, we train bidirectional neural language models using sensor names from both source and target buildings; we expect such co-training to regularize the feature learning process for our tagger so that the model can be better applied to the target building. In addition, we propose to learn k-mer (i.e., a substring of length-k) representations of the source building and align them with those of the target building, as there exist common character patterns across buildings similar to "words" in human language. For example, "`T`" or "`temp`" would almost always appear in sensors related to temperature. These aligned k-mers complement the language model as "word"-level information, namely, what phrases look like in sensor names. Moreover, during inference, because of a strong connection between raw names and entity types, we incorporate information (e.g., what an abbreviate stands for) obtained from resources such as Wikipedia as prior knowledge to narrow the gap between the limited input data and a large number of target classes.

In summary, our major contributions are:

- We study an important problem of exploiting existing annotated buildings to help train a sensor metadata tagger for a new building.
- We propose a novel framework, SeNsER, which leverages neural language models to regularize the feature learning process and utilizes k-mers from the source building to help annotate the target building, aided by prior knowledge extracted from sources such as Wikipedia.
- We conduct extensive experiments on real buildings consisting of thousands of sensor names. SeNsER achieves over 79% and 67% F_1 in chunking and tagging, respectively – a notable 13-point improvement in tagging over the best compared method.

Reproducibility. We release our code and datasets on GitHub [1].

2 Related Work

We review the literature from two fields, namely, sensor name tagging and named entity recognition.

Sensor Metadata Tagging. The problem of tagging sensor metadata has seen increasing interest from the smart building and sensing communities, mainly following the active learning paradigm (Settles, 2009) to reduce manual labeling effort. These methods iteratively select "representative" metadata examples for a human to annotate and progressively craft custom regular expressions (Bhattacharya et al., 2015) or construct classical learning models such as logistic regression (Hong et al., 2015b; Ma et al., 2020) and conditional random fields (Balaji et al., 2015; Koh et al., 2018; Lin et al., 2019), in order to tag the sensor names. Despite the promising results, all these methods rely on building-specific domain knowledge and human effort, which often do not generalize across buildings.

Another attempt based on transfer learning (Hong et al., 2015a) leverages the information from existing buildings to classify sensor measurement type only, which is a sub-problem of sensor tagging. It is primarily built upon sensory time-series data and therefore cannot generalize to other contextual information, such as the location and relationship with others. By contrast, we aim to understand all the information in the metadata.

Named Entity Recognition (NER). Our sensor metadata tagging problem can be viewed as a kind of NER task, while our tagging happens *per character*. Most of, if not all, NER models consume words as the basic unit and detect entity boundaries as a subset of word boundaries. However, in our problem, due to the lack of pre-processing tools, the input only contains raw character sequences. Such difference makes most of the recent neural NER models (Peters et al., 2018; Devlin et al., 2018; Akbik et al., 2018; Huang et al., 2015; Lample et al.,

[1] https://github.com/JiachengLi1995/SeNsER

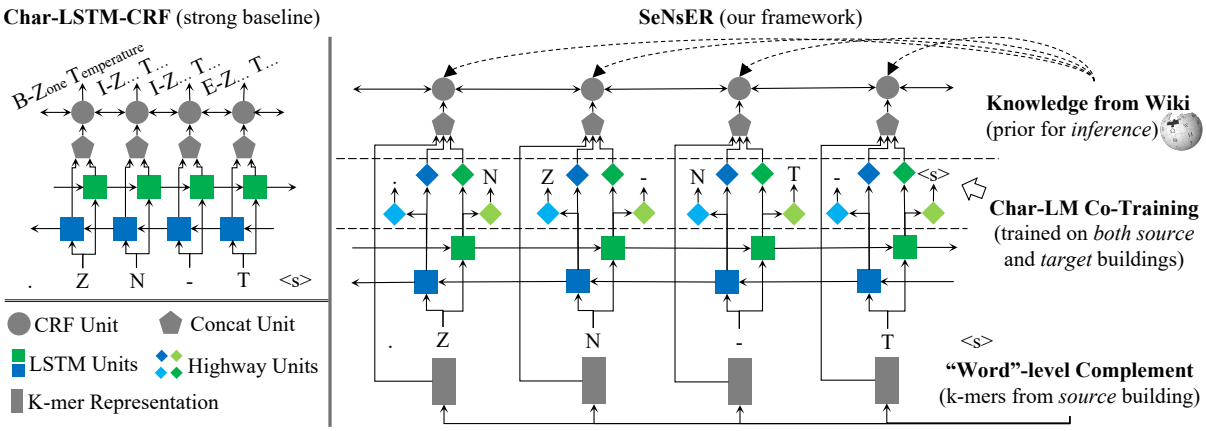

Figure 2: Neural architecture of SeNsER. Built upon Char-LSTM-CRF, SeNsER further equips it with three novel components for effective cross-building metadata tagging: (1) *co-training* of language models using both source and target buildings to guide feature learning, (2) *k-mer*-based "word"-level information to assist with alignment, and (3) *prior knowledge* obtained from external sources such as Wikipedia to help inference.

2016; Ma and Hovy, 2016; Liu et al., 2018b; Kuru et al., 2016) not directly applicable. After sifting through compatible modules from these models, the best applicable existing neural NER model becomes Char-LSTM-CRF, as we shall describe in Section 4. It performs well under the intra-building setting but poorly under our cross-building setting.

Our idea of introducing language models as regularization is inspired by LM-LSTM-CRF (Liu et al., 2018b). LM-LSTM-CRF imposes a language model objective on the NER's training set as additional guidance for feature extraction. In this paper, we further propose to train the language models using *both* source and target buildings, so as to better generalize the features learned from the source building to the target building.

3 Problem Formulation

In this paper, we study the cross-building metadata tagging problem. The input involves two buildings: (1) a fully annotated source building, and (2) a new target building with no annotation. The metadata of a sensor is a sequence of characters, denoted as $X = (x_1, x_2, \ldots, x_M)$, where x_i $(1 \leq i \leq M)$ is the i-th character and M is the length of the sequence. We denote the annotation of token x_i as y_i. Similar to NER, the annotations follow the `BIOES` labeling scheme (Ratinov and Roth, 2009), but at the *character level*. We define a segment of sensor name to be a substring expressing certain context (e.g., building name, room, measurement type, etc) about the sensor, as illustrated in Figure 1. Given a segment in the sensor name, its beginning, middle, and ending characters are labeled as `B-type`, `I-type`, and `E-type`, respectively. Segments with only one character are labeled as `S-`, and characters not belonging to any segment will be marked as `O`. All `BIES` labels are followed by a particular class. It is noteworthy that there are more than 100 classes for the different segments in sensor names, such as `building name`, `room`, `heating` (a sensor type), etc. Our goal is to learn a tagger for the target building, which can partition the sensor name into correct segments and classify them into the right classes.

4 Char-LSTM-CRF

As mentioned earlier, the best applicable existing neural tagging model to our problem is Char-LSTM-CRF, which was proposed as a strong baseline in (Liu et al., 2018a). Since SeNsER builds upon Char-LSTM-CRF, we briefly revisit this model to be self-contained.

As illustrated by the top part of Figure 2, Char-LSTM-CRF takes as input a character sequence $X = (x_1, x_2, \ldots, x_M)$, and applies bidirectional LSTMs to every character's embedding, obtaining $\mathbf{f}_i$ and $\mathbf{r}_i$ for the i-th character. Then, it gets the contextualized representation $\mathbf{z}_i$ of the i-th character by concatenating the two embedding vectors:

$$\mathbf{z}_i = [\mathbf{f}_i; \mathbf{r}_i].$$

Finally, it uses a Conditional Random Field (CRF) layer (Lafferty et al., 2001) to capture the label

dependency, which defines the probability of generating the label sequence $Y = (y_1, y_2, \ldots, y_M)$, namely,

$$P(Y|Z) = \frac{\prod_{j=1}^{M} \phi(y_{j-1}, y_j, \mathbf{z}_j)}{\sum_{\hat{Y} \in \mathcal{Y}(Z)} \prod_{j=1}^{M} \phi(\hat{y}_{j-1}, \hat{y}_j, \mathbf{z}_j)},$$

where $\mathcal{Y}(Z)$ is the set of all possible label sequences, $\phi(y_{j-1}, y_j, \mathbf{z}_j) = \exp(\mathbf{W}_{y_j}\mathbf{z}_j + \mathbf{b}_{y_{j-1},y_j})$, and $\mathbf{W}_{y_j}$ and b_{y_{j-1},y_j} are the weight and bias parameters in the CRF layer, respectively.

During training, we maximize the likelihood of generating the ground-truth label sequences, hence the following loss function:

$$\mathcal{L}_{CRF} = -\sum_i \log P(Y^i|Z^i),$$

where Y^i is the label sequence and Z^i is the embedding for the i-th training example (i.e., sensor name). For inference, we use the Viterbi algorithm (Viterbi, 1967) to decode the best explanation given Z.

5 Our SeNsER Framework

As shown in Figure 2, our SeNsER framework builds upon Char-LSTM-CRF and further enhances it with (1) cross-building language models as regularization, (2) k-mer alignment as "word"-level complement, and (3) tailored decoding using a domain-specific dictionary as prior knowledge.

5.1 Language Models as Regularization

To address the heterogeneity between the source and target buildings, we propose to co-train the character-level neural language models (Char-LMs) on the raw sensor names from *both* buildings in addition to the tagging objective. Here, "co-training" means that the LSTM modules are shared between our bidirectional Char-LMs and the Char-LSTM-CRF tagging model, and that their parameters will be updated by two objectives together. We shall note that we only incorporate the raw sensor names, but not their labels for a target building in training the language model (Char-LMs). This way, the LSTM modules will also be regularized by the raw sensor names in the target building, significantly improving generalizability when we apply the trained tagger to the target building.

The forward Char-LM defines the generative probability of a character sequence as

$$P_{fw}(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P_{fw}(x_i|x_1, \ldots, x_{i-1}).$$

Denoting the representation after reading $x_1, \ldots, x_i$ in the forward Char-LM as $\mathbf{f}_i^{LM}$, $P_{fw}(x_i|x_1, \ldots, x_{i-1})$ can be written as

$$P_{fw}(x_i|x_1, \ldots, x_{i-1}) = P_{fw}(x_i|\mathbf{f}_{i-1}^{LM}).$$

We apply softmax to $\mathbf{f}_{i-1}^{LM}$ to obtain this probability. Inspired by previous work (Liu et al., 2018b), we adopt a highway layer to further introduce nonlinear transformation from $\mathbf{f}_i$ to $\mathbf{f}_i^{LM}$:

$$\mathbf{f}_i^{LM} = H(\mathbf{f}_i) = \mathbf{t} \odot g(\mathbf{W}_H\mathbf{f}_i + \mathbf{b}_H) + (\mathbf{1} - \mathbf{t}) \odot \mathbf{f}_i,$$

where $\odot$ is element-wise product, $g()$ is a nonlinear transformation such as ReLU in our experiments, $\mathbf{W}_H$ and $\mathbf{b}_H$ are two parameters in the highway layer, and $\mathbf{t} = \sigma(\mathbf{W}_H\mathbf{f}_i + \mathbf{b}_T)$ is called transform gate and $(\mathbf{1} - \mathbf{t})$ is called carry gate. Here $\sigma()$ is some nonlinear function such as sigmoid.

Similarly, one can define $\mathbf{r}_i^{LM}$ and $P_{bw}(x_i|\mathbf{r}_{i+1}^{LM})$. Adding the two directions together, the loss function of the language model part becomes:

$$\mathcal{L}_{LM} = -\sum_i \left(\log P_{fw}(x_i|\mathbf{f}_{i-1}^{LM}) + \log P_{bw}(x_i|\mathbf{r}_{i+1}^{LM}) \right).$$

The contextualized representation $\mathbf{z}_i$ of character x_i is also revised accordingly. The $\mathbf{f}_i$ and $\mathbf{r}_i$ are passed through two high-way units and become $\mathbf{f}_i^H$ and $\mathbf{r}_i^H$, respectively. Now, after enabling this co-training, it becomes:

$$\mathbf{z}_i = [\mathbf{f}_i^H; \mathbf{r}_i^H]. \quad (1)$$

Joint Optimization. We jointly optimize the Char-LSTM-CRF and Char-LM via

$$\mathcal{L} = (1 - \lambda)\mathcal{L}_{CRF} + \lambda\mathcal{L}_{LM},$$

where $\lambda \in [0, 1]$ is a weight balancing the effect of Char-LSTM-CRF and Char-LM on training. To ensure the model is not overfitted in the source building, in practice, we always start with $\lambda = 1$ and linearly decrease it as the training proceeds.

5.2 K-Mers as "Word"-level Complement

So far, SeNsER is solely built upon character-level information. We observe that some k-mers (i.e., substrings of length-k) (Compeau et al., 2011) express the same meaning regardless of buildings, e.g., "`T`", "`tmp`", or "`temp`" almost always appear in sensor names related to temperature. Therefore, we propose to leverage such meaningful k-mers to complement the representation produced by the language model (i.e., $\mathbf{z}_i$ defined in Eq. (1)) as "word"-level information.

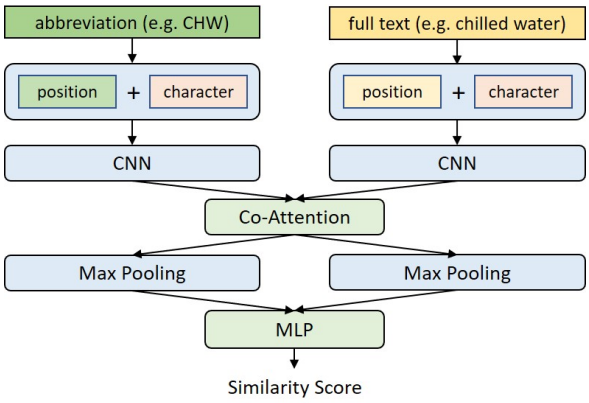

Figure 3: Our Siamese Network for Abbreviation-Phrase Matching Model.

Specifically, in the source building, we obtain a k-mer vocabulary using the sensor names and their annotations – every ground-truth segment in a sensor name becomes a k-mer. We then apply word embedding techniques (e.g. word2vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014)) to learn representations of these k-mers. During training in the source building, we use its annotations to align every character with the k-mer it appears in. During inference in the target building, we match a raw sensor name string to the k-mers in the vocabulary by trying to cover as many characters as possible with the most informative k-mer combinations, where each k-mer is scored by its inverse "document" frequency (IDF); in our context, a "document" is a sensor name. Given a sensor name, for its character x_i, after aligning it with a k-mer we incorporate this k-mer representation $\mathbf{k}_i$ into $\mathbf{z}_i$, i.e.,

$$\mathbf{z}_i = \left[\mathbf{f}_i^H; \mathbf{r}_i^H; \mathbf{k}_i\right].$$

Through a dynamic programming algorithm, we can partition a raw string in the target building into pieces and maximize the total IDF of each k-mer. Characters that fail to be matched in this way will be matched to "<unk>".

5.3 Inference with Domain Knowledge

In order to accommodate more than 100 tagging types, given the fact that a certain segment of a sensor name indicates its measurement type, we propose to develop a domain-specific abbreviation-phrase matching model and employ it as *additional prior* during CRF decoding. We next discuss (1) how to build this matching model, and (2) how to incorporate it into the CRF layer.

Abbreviation-Phrase Matching. To get the most likely abbreviations of the type phrases, we propose a new character-level text similarity model based on Siamese Network (Bromley et al., 1993). The structure of the similarity model is depicted in Figure 3. Specifically, type phrases and abbreviations are embedded into a common latent space considering both the characters and their absolute positions. Then, we apply two 1D convolutional neural networks (CNNs) to encode the context. To capture the mutual information between two sentences, we adopt the co-attention idea (Ye and Ling, 2019) and apply it at the character level. After max-pooling, we get the final representations for the type phrase and abbreviation. We feed the concatenation of these two representations into a Multi-Layer Perceptron (MLP) with nonlinear activation to get a similarity score.

In order to train this model, we scraped a domain-specific abbreviation dataset from Wikipedia and technical documents in the building domain, which contains 574 abbreviations and 737 full names. We split the dataset into train, validation, and test sets with a 80%-10%-10% ratio of abbreviations, and 1:1 positive-negative pairs are sampled during training and testing. Following a prior work on learning text similarity (Neculoiu et al., 2016), we adopt the contrastive loss function for training. As binary classification evaluation (0.5 as a threshold), our trained model can on average achieve 98% test accuracy in matching an abbreviation to the full phrase, demonstrating its efficacy. Finally, we train a model on the entire abbreviation dataset we scraped and then obtain a set of potential tagging labels for each abbreviation with corresponding similarity scores.

We release our code and dataset for abbreviation-phrase matching on Github[2].

Additional Prior in CRF Decoding. In order to assign each character x_i a similarity score, we conduct a substring search around it to assign it to an associated abbreviation. Specifically, we check all the substrings within ± 2 positions around x_i (inclusive), i.e., all substrings of $x_{[i-2:i+2]}$, and check the similarity between these substrings and different tagging labels. The longest and most similar substring match will be assigned as the associated abbreviation for x_i. The similarity scores between this abbreviation and tagging labels are then propagated to $\text{sim}(x_i, y_i)$. We incorporate this similarity

[2] https://github.com/JiachengLi1995/Character-level-text-similarity

into the CRF decoding stage as follows:

$$P(Y|Z)=\frac{\prod_{j=1}^{M}\phi(y_{j-1},y_j,\mathbf{z}_j)\cdot\text{sim}(x_i,y_i)}{\sum_{\hat{Y}\in\mathcal{Y}(Z)}\prod_{j=1}^{M}\phi(\hat{y}_{j-1},\hat{y}_j,\mathbf{z}_j)\cdot\text{sim}(x_i,y_i)}.$$

The Viterbi algorithm (Viterbi, 1967) still applies without any computational overhead.

6 Empirical Evaluation

In this section, we empirically evaluate SeNsER and compared models on real-world buildings. We first introduce the datasets and experimental settings. Then, we present chunking and tagging results. Finally, we present some case studies about k-mer embedding and typical mistakes of our model.

6.1 Datasets

To evaluate SeNsER, we collect the sensor names from three office buildings on two different campuses, and the building names are anonymized as **A**, **B**, and **C**. The ground-truth labels of sensor names are created by the building vendors, which we subsequently convert to the character-level `IOBES` labels. The details of each building are summarized in Table 1.

Buildings A and B are on the same campus contracted with the same vendor, thus exhibiting similar naming conventions; yet their sensor names still contain unique tags due to different sensors and equipment deployed, and variations also exist even in the "codes" used for the same type of sensors, as illustrated in Table 1. Since it is impossible for the model to predict for classes out of the training set, we thus only keep the overlapping classes between a pair of source and target buildings in evaluation. In other words, given a pair of buildings, if a class exists only in either of the two buildings, we will mark it as an "other" class. As a result, a total of 70 classes, consisting of 69 regular classes and one "other" class, remain in our experiments between buildings A and B.

Building C is located on a second campus and is commissioned by a different vendor than A and B's; we use it to examine the generalizability of our method. There are only 4 classes in building C that appear in either building A or B, so there is not much difference between chunking and tagging. Therefore, we only evaluate chunking when training models based on building A and B and testing them on building C.

6.2 Metrics and Compared Methods

We evaluate the performance of SeNsER with regard to chunking and tagging using the precision, recall, and F_1 scores, similar to NER tasks. Specifically, for each sensor name, we get a few predicted triplets, i.e., $(position_{begin}, position_{end}, category)$, and only when both the position and category exactly match the ground-truth annotations does it count as a correct extraction. For chunking, we only consider the positions. Mathematically, we compare two sets of triplets, i.e., the predicted set and the ground-truth set. True positive is the intersection between the two sets. The remaining triplets in the predicted and ground-truth sets are considered as false positive and false negative, respectively.

We compare SeNsER with the following methods as baselines:

- **CRF**. As the most straightforward baseline, we compare SeNsER with a standard CRF which is trained on the source building and applied to the target building. Particularly, 6 features are used in total, including *is* x_i *a digit*, *is* x_i *a letter*, *is* $x_{i\pm1}$ *a digit*, *is* $x_{i\pm1}$ *a letter*.
- **Char-LSTM-CRF**. As we described in Section 4, it first applies bidirectional LSTMs to every character's embedding and further feeds it into the CRF layer, and finally outputs labels for each character.

As a sanity check, we also examine two methods:

- **Delimiter**. Sensor names usually contain delimiters such as "–" and "`.`". Therefore, as a straightforward option for *chunking*, we segment sensor names at the positions of delimiter and then calculate the precision, recall, and F_1.
- **Dictionary (Dict)**. For this method, we use the dictionary created in §5.3 and decode the type of label using the Viterbi algorithm.

We also evaluate ablations of our model. **SeNsER-Dict** only keeps the use of the dictionary comprised of abbreviation-phrase pairs during inference by removing the k-mer alignments from SeNsER, and likewise, **SeNsER-Kmer** keeps only k-mer alignments by removing the use of the dictionary. We shall note that, technically, Char-LSTM-CRF is also the ablated version of SeNsER with none of the proposed components used, namely, co-training, k-mers matching, and dictionary as prior.

We only use Char-LSTM-CRF as our NER

Table 1: Statistics of the buildings from two campuses used in our experiments. Buildings A and B are from the same campus, while C is on a different one. We also present example names for *room temperature sensor* in the three buildings used in our study: variations exist even in buildings (A and B) by the same vendor.

Building	#Sensors	#Classes	Name Length	Sensor Name
A	4,954	157	$11 \sim 14$	EBU3B.RM-B215..ZN-T
B	4,357	134	$10 \sim 17$	AP&M.RM-1011.TEMP
C	2,551	63	$7 \sim 31$	SDH_SF1_R282_RMT

Table 2: Cross-building tagging and chunking performance (%). "X $\rightarrow$ Y" denotes to train a tagger on building X and test on building Y. All results are averaged over 5 runs. We omit the standard deviations as they are all $\leq 2\%$.

	Building A $\rightarrow$ Building B						Building B $\rightarrow$ Building A					
	Chunking			Tagging			Chunking			Tagging		
Methods	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1	Precision	Recall	F_1
Delimiter	54.10	34.65	42.24	-	-	-	46.24	31.94	37.78	-	-	-
Dict	45.61	14.11	21.56	36.12	10.88	16.73	30.41	7.18	11.62	25.10	5.89	9.54
CRF	58.09	58.32	58.12	44.35	55.31	49.13	58.75	60.32	59.44	37.63	38.95	38.26
CRF-Kmer	58.12	81.47	67.59	52.41	69.57	59.82	50.53	40.46	44.80	47.92	34.80	41.62
CRF-Dict	73.64	74.81	74.11	68.10	70.50	69.18	65.70	61.56	63.56	54.47	47.93	50.99
Char-LSTM-CRF	84.29	75.31	79.54	65.48	58.27	61.66	**79.82**	74.13	**76.86**	43.47	51.89	47.37
SeNsER-Kmer	73.22	77.57	75.25	61.19	73.73	66.87	63.02	62.62	62.81	58.29	53.84	55.81
SeNsER-Dict	86.26	89.14	87.68	**70.60**	71.14	70.87	67.77	73.03	70.30	**64.80**	56.42	60.32
SeNsER	**86.81**	**89.52**	**88.15**	66.84	**77.89**	**71.78**	66.08	**76.35**	70.70	59.72	**67.27**	**62.55**

model because we study char-level tagging with limited training data. Other models (e.g., BERT) typically require large-scale corpus for pretraining and word-level input, which are not available in the building domain we study.

Regular expression (regex) could be a solution to our problem, but they need to be exhaustive in covering all the possible patterns, which requires deep building-specific domain knowledge and significant manual effort at great costs to create on a per-building basis. Moreover, regex for tagging patterns cannot transfer across buildings, which is our goal in this work. Therefore, regex is neither an economical nor scalable solution.

For a fair comparison, all baselines use the same amount of human labels. Because of the considerable amount of human effort needed for regexes, we do not include it for comparison in this work. The **Delimiter** method can be viewed as a special kind of regexes, with a minimum amount of human effort.

6.3 Experimental Setup

During training, 80% of the sensor names in the source building are used as the training set and the remaining 20% is used as a development set; testing is performed on the sensor names in the target building. Mini-batch stochastic gradient descent with momentum is used for training all the neural models. For all three models, the batch size, momentum, and learning rate are set to 10, 0.9 and $\eta_t = \frac{\eta_0}{1+\rho_t}$, where η_0 is the initial learning rate and $\rho = 0.05$ is the decay ratio. We apply dropout with a ratio of 0.5. Models are trained for a maximum of 200 epochs, and early stop happens when the current best F_1 score on development set does not increase for 15 epochs.

The dimension of randomly-initialized character embedding and character-level LSTM state is set to 30 and 150, respectively. For word embedding of k-Mers, we apply word2vec (Mikolov et al., 2013) on these "words" and the dimension of embedding is set to 30. Other word embedding techniques (e.g., Glove (Pennington et al., 2014)) also work for this part. In language models, we set the dimension of LSTM state to 300. For the parameter λ, which balances the effect of Char-LM and Char-LSTM-CRF during training, it is initialized to 1 and decreases along the training process until it reaches a particular minimum value. This way, during the multi-task training of Char-LM and Char-LSTM-CRF, the model in the early epochs will focus more on learning an effective LM for understanding the sequence characteristics, which benefits the learning of Char-LSTM-CRF in the later stage of training and transfer learning.

Table 3: Top-5 similar k-mers to query k-mer based on its embedding.

	k-mer	Explanation	k-mer	Explanation	k-mer	Explanation
Query	htg	heating	co	CO_2	ef	exhuast fan
1	clg	cooling	level	level	ahu	air handling unit
2	sup	supply	dasp	discharge air setpoint	e	exhaust air
3	vp	velocity pressure	box	box	st	status
4	rh	reheat	vp	velocity pressure	cm	command
5	enbl	enable	ll	low limit	speed	speed

Table 4: Chunking performance on building C of tagger trained using data from both building A and B.

	Precision	Recall	F_1
Delimiter	49.31	27.48	35.29
Dict	39.54	11.42	17.73
CRF	79.19	66.54	72.32
Char-LSTM-CRF	56.68	50.75	53.49
SeNsER	78.96	77.44	78.18

6.4 Cross-Building Performance

We summarize the cross-building chunking and tagging performance in Table 2. In general, our experimental results suggest that transferring from A to B is better than B to A. The main reason is that building A contains more types of metadata labels (i.e., 157) than building B (i.e., 134). SeNsER would be more effective if trained on a dataset with various sensors and applied to a dataset with relatively fewer types of sensors.

Besides, we observe that the majority of correct chunks obtained by the delimiter method are the building names, which appear almost in all the metadata sequences at the beginning, followed by a delimiter. However, room or floor segments usually contain delimiters such as "_" and "–", and thus will be incorrectly segmented by this method. As a sanity check, we also directly apply the dictionary built upon online documents such as Wikipedia, which consists of abbreviation codes used in the building domain. As the dictionary is not exhaustive, solely matching based on the abbreviations in the dictionary can only uncover a small fraction of segments, hence the limited chunking and tagging results.

As a common solution to NER, CRF with handcrafted features achieves decent chunking results (58.78% F_1 on average), yet struggles with tagging (43.70% F_1 on average), since the "codes" used in building A and B vary. To demonstrate the efficacy of the proposed k-mer-based alignments and dictionary as prior knowledge, we also incorporate them into the standard CRF as CRF-Kmer and CRF-Dict. As we see from the results, both can enhance a standard CRF in chunking and tagging.

Char-LSTM-CRF significantly improves over CRF by learning the features to represent the generative pattern in sensor names, achieving 78.20% and 54.52% on average in F_1 for chunking and tagging, respectively. Compared to Char-LSTM-CRF, SeNsER-Kmer additionally employs the k-mer-based alignment procedure to help identify segments in sensor names in the target building. We see that it improves tagging by 6.83 points in F_1 on average. On another front, SeNsER-Dict incorporates as prior knowledge during inference the dictionary of abbreviations-phrases pairs. Similar to what we have observed for the case of CRF, this knowledge clearly benefits both chunking and tagging on the two buildings. Finally, employing both the k-mer alignments and dictionary of abbreviation-phrase pairs, SeNsER considerably outperforms the best baseline by 8.61 points in chunking on building B, and by an average 12.65 points in tagging on both buildings.

The superior performance of SeNsER confirms the synergy between language models for capturing contextual information and k-mers for substring alignments in different buildings as well as a dictionary as prior knowledge (especially with a limited vocabulary).

6.5 Case Study

Similar K-mers. K-mers have demonstrated their power in recognizing the class of name segments, i.e., tagging. Here, we present a case study about the learned k-mer embedding results. It will provide some insights into the usefulness of our k-mer

alignment. In Table 3, we present three random k-mers from our vocabulary and retrieve their top-5 similar words according to cosine similarity. The results are reasonable, containing semantically correlated k-mers. For example, `heating` equipment commonly pairs with the corresponding `cooling` equipment to condition a room/zone, and involves `reheating`, measurements of `supply` airflow, and `velocity pressure`.

Typical Mistakes. The most common mistakes in our inference occur in the building name segments. Our SeNsER can effectively learn the common features of different buildings such as temperature and equipment operating status. However, the building names vary a lot in different buildings and share no similar features; for example, recall the examples in Table 1, the building name phrases are `EBU3b`, `ap&m`, and `SDH`. Without human input, it is difficult for our model to correctly infer the meaning of such segments. However, as a possible future direction to pursue, based on the frequency, we could infer with a high probability that a segment is likely to be the building name, and therefore query a human for a one-time input to label all such segments.

6.6 Generalizability

We also examine the generalizability of our method, i.e., how it would perform when applied to a building with a completely distinct vocabulary and naming convention. In particular, we train a tagger using the sensor names and annotations in building A and B and apply it to building C.

Note that, this is an extremely difficult task: Building A and B still share similar naming conventions (recall the examples in Table 1), despite moderately varied vocabulary; however, by contrast, building C almost completely differs in the naming convention and vocabulary. For example, "`room temperature`" is denoted as "`ZN.T`" in A and B but as "`RMT`" in C; in addition, due to the different vendors used, the types of equipment installed also vary significantly in Building C, compared to Building A and B. Due to the disparate vocabularies, tagging Building C based on the information in A and B is nearly impossible, and we thus only take the prefixes of the tags produced by the tagger (i.e., `B-`, `I-` prefixes) to evaluate the chunking results.

The results are summarized in Table 4. Delimiter-based chunking method can achieve 35.29% in F_1, with the hits mainly being the first segments of the metadata string denoting building names, which do not vary in the building. It is noteworthy that Char-LSTM-CRF performs worse than CRF, which indicates that learning solely based on data from buildings A and B may even hurt the performance on building C. SeNsER is able to score a 78.18% F_1, best among all, in spite of the distinction between the source and target. Upon closer inspection, due to the employed Char-LMs, SeNsER can recognize the segments for sensor types and room IDs correctly.

7 Conclusions and Future Work

In this paper, we study the problem of automated cross-building sensor metadata tagging, a key to enabling any smart building applications. Capitalizing on the intuition that sensor names are created following some underlying rule, though varying across buildings, we design SeNsER. SeNsER builds upon Char-LSTM-CRF and guides the sensor name feature learning using both source and target buildings, well preparing them for interpreting the metadata in the target building. We further leverage a k-mer-based matching procedure to provide "word"-level information, as well as a dictionary comprised of prior knowledge about sensor names, to boost the tagging performance. Promising experimental results demonstrate the synergy among neural language models, k-mers-based alignments, and the use of prior knowledge.

As future work, we plan to further collect more domain-specific text data, e.g., sensor datasheets, which helps provide more information about different naming conventions and abbreviations. We then can integrate such information into our model to make it generalize better.

Acknowledgement

We thanks reviewers for the anonymous comments and suggestions to improve this work. This work was supported in part by National Science Foundation 1940291 and CA-2040727. Any opinions, findings, and conclusions or recommendations expressed herein are those of the authors and should not be interpreted as necessarily representing the views, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright annotation hereon.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Bharathan Balaji, Chetan Verma, Balakrishnan Narayanaswamy, and Yuvraj Agarwal. 2015. Zodiac: Organizing large deployment of sensors to create reusable applications for buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 13–22. ACM.

Arka A Bhattacharya, Dezhi Hong, David Culler, Jorge Ortiz, Kamin Whitehouse, and Eugene Wu. 2015. Automated metadata construction to support portable building applications. In *Proceedings of the 2nd BuildSys*, pages 3–12. ACM.

Jane Bromley, James W. Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. *Int. J. Pattern Recognit. Artif. Intell.*, 7:669–688.

Phillip EC Compeau, Pavel A Pevzner, and Glenn Tesler. 2011. How to apply de bruijn graphs to genome assembly. *Nature biotechnology*, 29(11):987.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dezhi Hong, Hongning Wang, Jorge Ortiz, and Kamin Whitehouse. 2015a. The building adapter: Towards quickly applying building analytics at scale. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*, pages 123–132. ACM.

Dezhi Hong, Hongning Wang, and Kamin Whitehouse. 2015b. Clustering-based active learning on sensor type classification in buildings. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 363–372. ACM.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Jason Koh, Bharathan Balaji, Dhiman Sengupta, Julian McAuley, Rajesh Gupta, and Yuvraj Agarwal. 2018. Scrabble: transferrable semi-automated semantic metadata normalization using intermediate representation. In *Proceedings of the 5th Conference on Systems for Built Environments*, pages 11–20. ACM.

Onur Kuru, Ozan Arkan Can, and Deniz Yuret. 2016. Charner: Character-level named entity recognition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 911–921.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of NAACL-HLT*, pages 260–270.

Lu Lin, Zheng Luo, Dezhi Hong, and Hongning Wang. 2019. Sequential learning with active partial labeling for building metadata. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 189–192. ACM.

Liyuan Liu, Xiang Ren, Jingbo Shang, Xiaotao Gu, Jian Peng, and Jiawei Han. 2018a. Efficient contextualized representation: Language model pruning for sequence labeling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1215–1225.

Liyuan Liu, Jingbo Shang, Xiang Ren, Frank Fangzheng Xu, Huan Gui, Jian Peng, and Jiawei Han. 2018b. Empower sequence labeling with task-aware neural language model. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Jing Ma, Dezhi Hong, and Hongning Wang. 2020. Selective sampling for sensor type classification in buildings. In *2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 241–252. IEEE.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning text similarity with siamese recurrent networks. In *Rep4NLP@ACL*.

Jeffrey Pennington, R. Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the ACL*.

Burr Settles. 2009. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences.

Erik F Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 142–147. Association for Computational Linguistics.

Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2):260–269.

Weimin Wang, Michael R Brambley, Woohyun Kim, Sriram Somasundaram, and Andrew J Stevens. 2018. Automated point mapping for building control systems: Recent advances and future research needs. *Automation in Construction*, 85:107–123.

Zhi-Xiu Ye and Zhen-Hua Ling. 2019. Multi-level matching and aggregation network for few-shot relation classification. In *ACL*.