CitySpec: An Intelligent Assistant System for Requirement Specification in Smart Cities

Zirong Chen*, Isaac Li†, Haoxiang Zhang‡, Sarah Preum§, John A. Stankovic†, Meiyi Ma*

* Vanderbilt University † University of Virginia ‡ Columbia University § Dartmouth College

{zirong.chen, meiyi.ma}@vanderbilt.edu {il5fq, stankovic}@virginia.edu sarah.masud.preum@dartmouth.edu

Abstract—An increasing number of monitoring systems have been developed in smart cities to ensure that a city's real-time operations satisfy safety and performance requirements. However, many existing city requirements are written in English with missing, inaccurate, or ambiguous information. There is a high demand for assisting city policy makers in converting human-specified requirements to machine-understandable formal specifications for monitoring systems. To tackle this limitation, we build *CitySpec*, the first intelligent assistant system for requirement specification in smart cities. To create CitySpec, we first collect over 1,500 real-world city requirements across different domains from over 100 cities and extract city-specific knowledge to generate a dataset of city vocabulary with 3,061 words. We also build a translation model and enhance it through requirement synthesis and develop a novel online learning framework with validation under uncertainty. The evaluation results on real-world city requirements show that CitySpec increases the sentence-level accuracy of requirement specification from 59.02% to 86.64%, and has strong adaptability to a new city and a new domain (e.g., F1 score for requirements in Seattle increases from 77.6% to 93.75% with online learning).

Index Terms—Requirement Specification, Intelligent Assistant, Monitoring, Smart City

I. Introduction

With the increasing demand for safety guarantees in smart cities, significant research efforts have been spent toward how to ensure that a city's real-time operations satisfy safety and performance requirements [1]. Monitoring systems, such as SaSTL runtime monitoring [2], CityResolver [3], and STL-U predictive monitoring [4], have been developed in smart cities. Figure 1 shows a general framework of monitoring systems in smart cities. These systems are designed to execute in city centers and to support decision-making based on the verification results of real-time sensing data about city-states (such as traffic and air pollution). If the monitor detects a requirement violation, the city operators can take actions to change the states, such as improving air quality, sending alarms to police, calling an ambulance, etc.

The monitor systems have two important inputs, i.e., the real-time data streams and formal specified requirements. Despite that extensive research efforts have been spent toward improving the expressiveness of specification languages and efficiency of the monitoring algorithms, the research challenge of how to convert human-specified requirements to machine-understandable formal specifications has received only scant attention. Moreover, our study (see Section II) on over 1,500
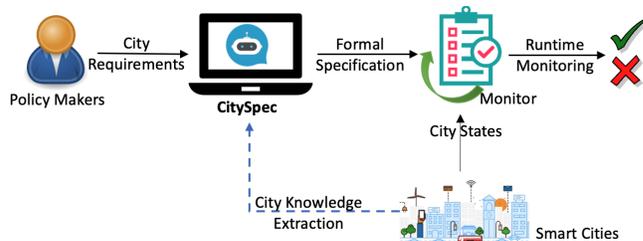

Fig. 1. CitySpec in Smart Cities

real-world city requirements across different domains shows that, first, existing city requirements are often defined with missing information or ambiguous description, e.g., no location information, using words like nearby, or close to. They are not precise enough to be converted to a formal specification or monitored in a city directly without clarifications by policy makers. Secondly, the language difference between English specified requirements and formalized specifications is significant. Without expertise in formal languages, it is extremely difficult or impossible for policy makers to write or convert their requirements to formal specifications. Therefore, there is an urgent demand for an intelligent system to support policy makers for requirement specifications in smart cities.

Despite the prevalence of developing models to translate natural language to machine languages in various domains, such as Bash commands [5], Seq2SQL [6], and Python [7], it is very **challenging** to develop such an intelligent system for requirement specification in smart cities for the following reasons. First, unlike the above translation tasks with thousands or even millions of samples in a dataset, there is barely any requirement specification data. As a result, traditional language models are not sufficient to be applied directly. Moreover, the requirements usually contain city domain-specific descriptions and patterns that existing pre-trained embeddings like BERT or GloVE cannot handle effectively. Furthermore, requirements from different domains and cities vary significantly and evolve over time, thus building a system that can adapt to new domains at runtime is an open research question. Good adaptability can increase user experience (e.g., policy makers do not have to clarify new terms repeatedly), while one of the major challenges is validating and filtering the new knowledge and avoiding adversarial examples online.

In this paper, we target the above technical challenges, and describe *CitySpec*, an intelligent assistant system for

requirement specification in smart cities. To the best of our knowledge, it is the first specification system helping city policy makers specify and translate their requirements into formal specifications automatically. As shown in Figure 1, CitySpec is designed to bridge the gap between city policy makers and monitoring systems. It enables policy makers to define their requirements by detecting missing, inaccurate, or ambiguous information through an intelligent assistant interface. To effectively train the translation model using a small amount of city requirement data, CitySpec extracts city knowledge and enhances the learning process through requirement synthesis. CitySpec can easily adapt to a new city or domain through online learning and validation.

Contributions. We summarize the major contributions of this paper as follows:

- We collect and annotate over 1,500 real-world city requirements from over 100 cities across different domains. We extract city-specific knowledge and build a dataset of city vocabulary with 3,061 words in 5 categories.
- We create an intelligent assistant system for requirement specification in smart cities. In the system, we build a translation model and enhance it through requirement synthesis, and develop a novel online learning framework with validation under uncertainty.
- We evaluate CitySpec extensively using real-world city requirements. The evaluation results show that CitySpec is effective on supporting policy makers accurately writing and refining their requirements. It increases the sentence level accuracy of requirement specification from 59.02% to 86.64% through city knowledge injection. It shows strong adaptability (user experience) to a new city (e.g., F1 score in Seattle from 77.6% to 93.75%) and a new domain (e.g., F1 score in security domain from 62.93% to 93.95%).

Paper organization: In the rest of the paper, we describe the motivating study of city requirement specification in Section II, provide an overview of CitySpec in Section III, and present the technical details in Section IV. We then present the evaluation results in Section V, discuss the related work in Section VI and draw conclusions in Section VII.

II. MOTIVATING STUDY

In this section, we study real-world city requirements and their formal specification as motivating examples to discuss the demand and challenges of developing an intelligent assistant system for requirement specification in smart cities. We collect and annotate over 1,500 real-world city requirements across different domains (e.g., transportation, environment, security, public safety, indoor environments, etc.) from over 100 cities. We make the following observations from the analysis of the requirement dataset.

Existing city requirements are often defined with missing information or ambiguous description. In [2], the authors define essential elements for monitoring a city requirement. Within the 1500 requirements, many requirements have one or more missing elements. For example, 27.6% of the requirements do

not have location information, 29.1% of the requirements do not have a proper quantifier, and 90% of the requirements do not have or only have a default time (e.g., always) defined. Additionally, requirements often have ambiguous descriptions that are difficult to be noticed by policy makers. For example, a location is specified as "nearby" or "close to". As a result, it is very difficult or impossible for the monitoring system to monitor these requirements properly. It indicates a high demand for an intelligent assistant system to support the policy makers to refine the requirements.

The language difference between English specified requirements and their formal specifications is significant. In Table I, we give three examples of city requirements in English, their formal specification in SaSTL, and the Damerau–Levenshtein Distance (DLD) [8] between each pair of requirements. DLD measures the edit distance between two sequences. It shows that natural languages are different from machine-compatible input languages. Formal specifications usually consist of mathematical symbols, which makes the conversion even more difficult. As shown in Table I, the average DLD from English requirements to formal specifications is 67, which means that it requires an average of 67 edits. As a reference, the average DLD brought by translating these three English requirements to Latin is 64.67. It indicates that the conversion from English requirements to formal specifications even requires more edits than the translation of these requirements from English to Latin. In general, building a translator from English to Latin would require millions of samples. However, as an under-exploited domain, there is a very limited number of well-defined requirements. Moreover, annotation of formal specifications requires specialties in formal methods and is extremely time-consuming. It presents major challenges for building such a translation model.

III. SYSTEM OVERVIEW

CitySpec is designed to bridge the gap between city policy makers and monitoring systems. It supports policy makers to precisely write city requirements in English through an intelligent interface, and then converts them to formal specifications automatically. An overview of CitySpec is shown in Figure 2. There are four major components in CitySpec, including an intelligent assistant *Interface* to communicate with policy makers (see Section IV-B), a *Requirement Synthesis* component to extract city knowledge and synthesize new requirements to build the translation model (see Section IV-C), a *Translation Model* to convert city requirements to formal specifications (see Section IV-D), and an *Online Learning* component to adapt the system to new knowledge (see Section IV-E).

At runtime (as indicated by the orange arrows in Figure 2), a city policy maker inputs a requirement in English through an intelligent assistant interface, which sends the requirements to the translation model. The *translation model* converts the requirements to a formal specification and checks if there is any missing information or ambiguous description. The translation model is built with injected city knowledge through requirement synthesis at the training time and enhanced

TABLE I

COMPARISON BETWEEN ENGLISH REQUIREMENTS AND FORMAL SPECIFICATIONS

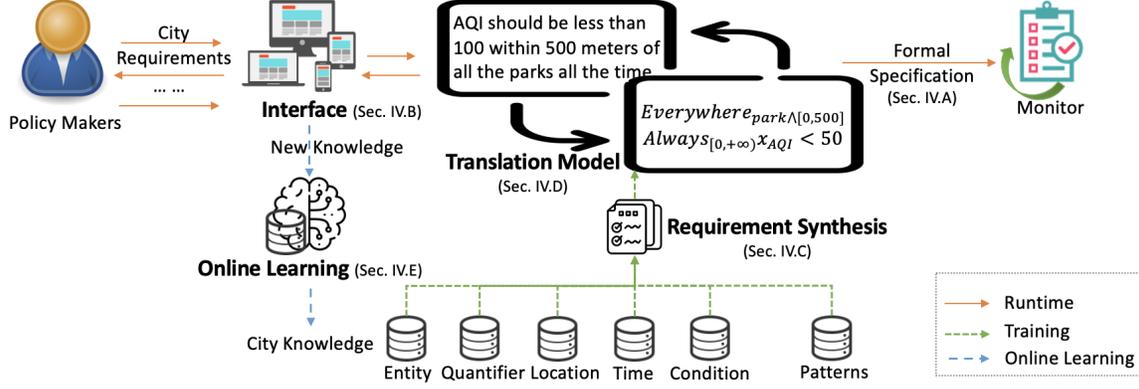| ID	English Requirement	Formal Specification	DLD
1	Sliding glass doors shall have an air infiltration rate of no more than 0.3 cfm per square foot.	$\text{Everywhere}_{\text{Sliding glass doors}}$ $\text{Always}_{[0,+\infty)}$ air infiltration rate $\leq$ 0.3 cfm/foot^2	59
2	The operation of a Golf Cart upon a Golf Cart Path shall be restricted to a maximum speed of 15 miles per hour from 8:00 to 16:00.	$\text{Everywhere}_{\text{Golf Cart Path}}\text{Always}_{[8,16]}$ Golf Cart speed < 15 miles/hour	67
3	Up to four vending vehicles may dispense merchandise in any given city block at any time.	$\text{Everywhere}_{\text{city block}}\text{Always}_{[0,+\infty)}$ vending vehicles $\leq$ 4	75

Fig. 2. System Overview

through online learning at runtime. Next, based on the returned results from the translation model, the intelligent interface communicates with the policy maker to acquire or clarify the essential information. In this process, the assistant supports the policy maker to refine the requirement until it is precisely defined and accepted by the monitor. We present the technical details in Section IV, and develop a prototype tool of the CitySpec system and deploy it online.

IV. METHODOLOGY

In this section, we present the major components in CitySpec (as shown in Figure 2). We first introduce requirement specification using Spatial-aggregation Signal Temporal Logic (SaSTL) [2]. Then we show the design and technical details of the intelligent assistant interface, requirement synthesis, translation model, and online learning, respectively.

A. Requirement Specification using SaSTL

SaSTL is a powerful formal specification language for Cyber-Physical Systems. We select it as our specification language because of its advantages of expressiveness and monitoring for smart cities. However, CitySpec is general and can work with other specification languages. SaSTL is defined on a multi-dimensional *spatial-temporal signal* as $\omega : \mathbb{T} \times L \to \{\mathbb{R} \cup \{\bot\}\}^n$, where $\mathbb{T} = \mathbb{R}_{\geq 0}$, represents the continuous time and L is the set of locations. $X = \{x_1, \cdots, x_n\}$ is denoted by the set of variables for each location. The spatial domain $\mathcal{D}$ is defined as, $\mathcal{D} := ([d_1, d_2], \psi)$, $\psi := \top \mid p \mid \neg \psi \mid \psi \vee \psi$, where $[d_1, d_2]$ defines a spatial interval with $d_1 < d_2$ and $d_1, d_2 \in \mathbb{R}$, and ψ specifies the property over the set of propositions that must hold in each location.

The syntax of SaSTL is given by

$$\varphi := \quad x \sim c \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \mathcal{U}_I \varphi_2 \mid \mathcal{A}_{\mathcal{D}}^{\text{op}} x \sim c \mid \mathcal{C}_{\mathcal{D}}^{\text{op}} \varphi \sim c$$

where $x \in X$, $\sim \in \{<, \leq\}$, $c \in \mathbb{R}$ is a constant, $I \subseteq \mathbb{R}_{>0}$ is a real positive dense time interval, $\mathcal{U}_I$ is the *bounded until* temporal operators from STL. The *always* (denoted $\square$) and *eventually* (denoted $\lozenge$) temporal operators can be derived the same way as in STL, where $\lozenge\varphi \equiv \text{true } \mathcal{U}_I\varphi$, and $\square\varphi \equiv \neg\lozenge\neg\varphi$. Spatial *aggregation* operators $\mathcal{A}_{\mathcal{D}}^{\text{op}} x \sim c$ for op $\in \{\max, \min, \text{sum}, \text{avg}\}$ evaluates the aggregated product of traces $\text{op}(\alpha_{\mathcal{D}}^x(\omega, t, l))$ over a set of locations $l \in L_{\mathcal{D}}^l$, and *counting* operators $\mathcal{C}_{\mathcal{D}}^{\text{op}}\varphi \sim c$ for op $\in \{\max, \min, \text{sum}, \text{avg}\}$ counts the satisfaction of traces over a set of locations. From *counting* operators, we derive the *everywhere* operator as $\boxdot_{\mathcal{D}}\varphi \equiv \mathcal{C}_{\mathcal{D}}^{\min}\varphi > 0$, and *somewhere* operator as $\Diamonddot_{\mathcal{D}}\varphi \equiv \mathcal{C}_{\mathcal{D}}^{\max}\varphi > 0$. Please refer to [2] for the detailed definition and semantics of SaSTL.

B. Interface for Intelligent Assistant

City requirements often have missing or ambiguous information, which may be unnoticed by policy makers. It leads to the demand for human inputs and clarification when converting them into formal specifications. Therefore, we design an intelligent assistant interface in CitySpec serving as an intermediary between policy makers and the translation model. It communicates with policy makers and confirms the final requirements through an intelligent conversation interface.

To briefly describe the communication process, users first input a requirement in English, e.g., "due to safety concerns, the number of taxis should be less than 10 between 7 am to 8 am". CitySpec interface passes the requirement to the translation model and gets a formal requirement ($\text{always}_{[7,8]}$number of taxi < 10) with the keywords including,

- entity: the requirement's main object, e.g., "the number",
- quantifier: the scope of an entity, e.g., "taxi",
- location: the location where this requirement is in effect, which is missing from the above example requirement,

- time: the time period during which this requirement is in effect, e.g., "between 7 am to 8 am",
- condition: the specific constraint on the entity, such as an upper or lower bound of entity, e.g., "10".

As a result, CitySpec detects that the location information is missing from the user's requirement and generates a query for the user, "what is the location for this requirement?" Next, with new information typed in by the user (e.g., "within 200 meters of all the schools"), CitySpec obtains a complete requirement.

The next challenge is how to confirm the formal specification with policy makers. Since they do not understand the formal equation, we further convert it to a template-based sentence. Therefore, CitySpec presents three formats of this requirements for users to verify, (1) a template-based requirement, e.g., [number] of [taxi] should be [<] [10] [between 7:00 to 8:00] [within 200 meters of all the schools], (2) a SaSTL formula everywhere$_{school \wedge [0,200]}$always$_{[7,8]}$number of taxi < 10, and (3) five key fields detected. Users can confirm or further revise this requirement through the intelligent assistant.

When policy makers have a large number of requirements to convert, to minimize user labor to input requirements manually, CitySpec also provides the option for them to input requirements through a file. The process is similar to where CitySpec asks users to provide or clarify information until all the requirements are successfully converted through files.

C. Requirement Synthesis

The amount of city requirement dataset is insufficient to train a decent translation model in an end-to-end manner. As we've discussed in Section I, it requires extensive domain knowledge in both city and formal specifications and is extremely time-consuming to annotate new requirements. Furthermore, a majority of the existing city requirements are qualitatively or imprecisely written, which cannot be added to the requirement dataset without refinement [2]. To mitigate the challenge of small data to build a translation model, we design a novel approach to incorporating city knowledge through controllable requirement synthesis.

There are two main reasons why converting a city requirement to a formal specification is challenging with a small amount of data. First, *the vocabulary of city requirements are very diverse*. For example, requirements from different cities (e.g., Seattle and New York City) or in different domains (e.g., transportation and environment) have totally different vocabulary for entities, locations, and conditions. Second, *the sentence structure (patterns) of requirements vary significantly when written by different people*. It is natural for human beings to describe the same thing using sentences.

Targeting these two challenges, we first extract city knowledge and build two knowledge datasets, i.e., a vocabulary set and a pattern set. The vocabulary set includes five keys of a requirement, i.e., entity, quantifier, location, time and condition. The pattern set includes requirement sentences with 5 keywords replaced by their labels. For instance, we have a requirement, "In all *buildings*/location, the average *concentration*/entity of *TVOC*/quantifier should be no more

---

Algorithm 1 Requirement Synthesis

Input: m set of keywords vocabularies $\{V_1, V_2 \ldots V_m\}$, Pattern P, synthesis index λ
Output: Set of requirements R
  Initialize R as an empty set$\{\}$
  Let $\ell = \lambda \cdot \max(|V_1|, |V_2|, \ldots, |V_m|)$
  for $i \in 1 \ldots m$ **do**
    Initialize S_i as an empty array $S_i = []$
    while $|S_i| < \ell$ **do**
      Create a random permutation of V_i: P = Permutate(V_i)
      Concatenate P to S_i: S_i = Concat(S_i, P)
    end while
  end for
  for $j \in 1 \ldots \ell$ **do**
    Combine keywords $S_1[j], S_2[j] \ldots S_m[j]$ with Pattern P to create a requirement r_j
    Add r_j to the set of requirements R
  end for
  return R

---

than *0.6 mg/m3*/condition for *every day*/time.", the pattern extracted is "In #location, the average #entity of #quantifier should be no more than #condition for #time."

We extract the knowledge set from city documents besides requirements so that we are not limited by the rules of requirements and enrich the knowledge of our model. For example, we extract 336 patterns and 3061 phrases (530 phrases in entity, 567 phrases in quantifier, 501 phrases in location, 595 phrases in condition, and 868 phrases in time).

Next, we designed an approach to synthesizing controllable requirement dataset efficiently. Intuitively, we can go through all the combinations of keywords and patterns to create the dataset of requirements, which is infeasible and may cause the model overfitting to the injected knowledge. In order to enhance the model's performance, we need to keep a balance between the coverage of each keyword and the times of keywords being seen in the generation. We denote λ as the synthesis index, which indicates the *minimum* number of times that a keyword appears in the generated set of requirements. Assuming we have m set of keywords vocabularies $\{V_1, V_2 \ldots V_m\}$ and a pattern set as P, we have the total number of synthesized requirements $\ell = \lambda \cdot \max(|V_1|, |V_2|, \ldots, |V_m|)$. For each set of vocabularies V_i, we first create a random permutation of V_i and repeat it until the total number of phrases reaches ℓ, then we concatenate them to an array S_i. Once we obtain $S_1, \ldots S_m$, we combine them with pattern P to generate a requirement set R. Refer to Algorithm 1 for more details.

D. Translation Model

The inputs of the translation model are requirements, and the outputs of this module are formal specifications with token-level classification. We implement the translation model with three major components, a learning model, knowledge injection through synthesized requirements, and keyword refinement.

To be noted, CitySpec does not build its own translation model from scratch. Instead, we tackle the limitation of the traditional language model and improve it for city requirement translation. Therefore, CitySpec is compatible with different language models.

In this paper, we implement four popular language models, which are Vanilla Seq2Seq, Stanford NLP NER, Bidirectional
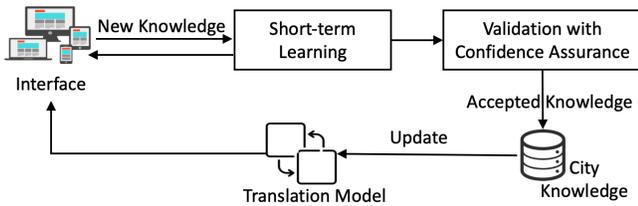
Fig. 3. Online Learning

Long Short Term Memory (Bi-LSTM) + Conditional Random Field (CRF) and Bidirectional Encoder Representations from Transformers (BERT) [9]. We apply our synthesized datasets with different synthesis indexes to inject city knowledge into these language models. Then we evaluate the improvement brought by our requirement synthesis approach by testing the performance on real-world city requirements. We present the detailed results and analysis in Section V.

Additionally, we find that *time, negation and comparison* are the most tricky elements that affect the accuracy of the final specification detection. Therefore, we implement another refinement component in the translation model. In general, the time domain can be represented in several formats, such as timestamps, or other formats like yyyy-mm-dd and mm-dd-yyyy. To mitigate the confusion that various formats might bring, we apply SUTime [10] when the time entity is not given by the translation model. pyContextNLP [11] is applied to analyze whether there is a negation in the input sentence. If there is any negation, the comparison symbol is reversed. For instance, if there is a keyword "greater than", the comparison symbol is >. However, if the whole phrase is "is not supposed to be greater than", and a negation is detected, thus the final comparison is ≤ instead.

E. Online Learning

In general, the more clarifications are needed from the users, the worse experience the users will have, especially if users have to clarify the same information repeatedly. For example, if a user from a new city inputs a location that the system fails to detect, the user will be asked to clarify the location information. The user's experience will drop if the system asks him again on the second or third time seeing these words. However, the deep learning-based translation model cannot "remember" this information at deployment time. Thus, the first question is that *how can CitySpec learn the new knowledge online?*

Meanwhile, the new information provided by users may also harm the system if it is an incorrect or adversarial example. The second question is that *how can CitySpec validate the new knowledge before learning it permanently?*

Targeting these two research questions, we design an online learning module in CitySpec. As shown in Figure 3, it has two stages, which are short-term learning and long-term learning. Short-term learning is designed to accommodate the same user in one session of requirement specification with a temporary memory. The question-answer pairs are stored temporarily. When the same occasion occurs, the temporal cache gives instant answers and avoids more user clarifications. Long-term learning is designed to adapt the new knowledge to the model permanently after validating its reliability. The accepted permanent knowledge is achieved by updating weights via back-propagation on the extended dataset with both initial data and the new input-label pairs stored in the temporary cache.

To keep CitySpec away from the adversarial inputs, we develop a Bayesian CNN-based validation module in CitySpec. The model is to classify the category of a new term with confidence with uncertainty estimation. We apply dropout layers during both training and testing to quantify the model uncertainty [4]. The inputs of the validation model are the new terms provided by the user, while the outputs are the corresponding keys among those five key domains with an uncertainty level. In brief, a new term-key pair is rejected if (1) the output from the validation function does not align with the given domain key; (2) the validation function has low confidence in the output although it might align with the given domain. In this way, we only accept new city knowledge validated with high confidence.

V. EVALUATION

In this section, we evaluate our CitySpec system from five aspects, including (1) comparing different language models on the initial dataset without synthesizing, (2) analyzing the effectiveness of the synthesized requirements by enhancing the models with city knowledge, (3) evaluating the performance of the online validation model, (4) testing CitySpec's adaptability in different cities and domains, and (5) an overall case study. We use the city requirement dataset described in Section II. To evaluate the prediction of keywords and mitigate the influence caused by different lengths requirements, we choose to use *token-level accuracy (token-acc)* and *sentence-level accuracy (sent-acc)* as our main metrics. The token-level accuracy aims to count the number of key tokens that are correctly predicted. The sentence-level accuracy counts the prediction as correct only when the whole requirement is correctly translated to a formal specification using SaSTL. Thus, sentence-level accuracy serves as a very strict criterion to evaluate the model performance. We also provide the results using other common metrics including precision, recall, and F-1 score. The experiments were run on a machine with 2.50GHz CPU, 32GB memory, and Nvidia GeForce RTX 3080Ti GPU.

A. Performance of language models on the initial dataset

As a baseline of translation model without city knowledge, we first evaluate the performance of CitySpec using different language models, including Vanilla Seq2Seq, pretrained Stanford NER Tagger, Bi-LSTM + CRF, and BERT on the initial dataset. We present the results in Table II.

We make the following observations from the results. First, the overlap between Stanford Pretrained NER Tagger prediction and vocabulary is only 9 out of 729. The pretrained tagger tends to give locations in higher granularity. Since this task is a city-level, more detailed location information is stated in a lower granularity by providing the street name, building name,

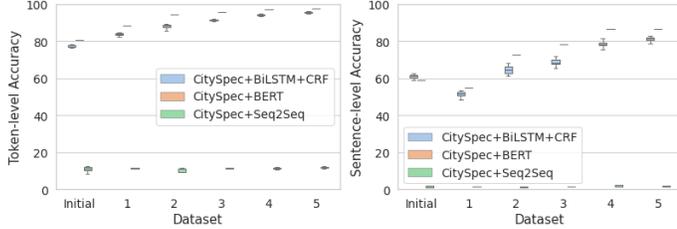| Model	Token-Acc	Sent-Acc	F-1 Score	Precision	Recall
Vanilla Seq2Seq	10.91 ± 0.57 %	1.38 ± 0.49%	24.12 ± 0.24 %	65.58 ± 7.95 %	14.81 ± 1.58 %
BiLSTM + CRF	77.59 ± 0.52 %	60.82 ± 1.22 %	80.46 ± 0.84 %	81.11 ± 1.38 %	79.83 ± 7.24 %
BERT	80.41 ± 0.07 %	59.02 ± 0.42 %	81.43 ± 0.01 %	78.62 ± 0.01 %	84.46 ± 0.01 %

Fig. 4. Performance improvement brought by requirement synthesis

or community name. The location domain in the pretrained tagger gives more high-level information like city name, state name, or country name. For example, "34th Ave in Nashville, the state of Tennessee" is annotated as location in this task, however, the pretrained NER tagger gives "Tennessee" as location instead.

Secondly, the testing token-acc from Vanilla Seq2Seq is 10.91% on average. Other metrics also indicate that Vanilla Seq2Seq has trouble recognizing the patterns in sequential keyword labeling. The Vanilla Seq2Seq model suffers from data scarcity and has difficulty recognizing the general patterns in the training samples due to the small size of the dataset.

Thirdly, the Bi-LSTM + CRF and BERT model achieve better performance than other models, and BERT models often outperform other models with lower standard deviation. However, the highest token-level accuracy achieved is 80.41%, which is still not high enough for an accuracy-prioritized task. A different key may change the requirement entirely. For example, the "width" of "car windshield" and the "width" of "car" focus on completely different aspects, although the keywords "car windshield" and "car" have an only one-word discrepancy. Meanwhile, the best sentence-level accuracy achieved is 60.82%, which means that about 40% of the requirements are falsely translated. Assuming the policy makers fix these requirements through the intelligent assistant interface, it is time-consuming and reduces the user experience. Even worse, it may bring safety issues to the monitoring system without noticing.

In summary, the results indicate that existing language models are not sufficient to serve as the translation model for CitySpec directly. There is a high demand for injecting city knowledge to build the translation model.

B. Requirement Synthesis with City Knowledge

Next, we evaluate CitySpec's performance with our controllable synthesized requirements. For a fair comparison, we do not use the requirements in the testing set to synthesize requirements. We ensure that the trained model has not seen the requirements in the testing set in either the knowledge injection or training phases. We apply different synthesis indexes to test the effects on the prediction performance. We present the overall results on token-level and sentence-level accuracy in Figure 4, and F-1 scores on individual keyword in Figure 5. In the figures, x-axis represents the synthesis index. When the index equals to "inital", it shows the results without synthesis data.

From the results, we find that, for BERT and Bi-LSTM, there is an overall increase in performance in all token-level accuracy, sentence-level accuracy, overall F-1 score, and F-1 score on keywords. For example, BiLSTM+CRF's token-level accuracy increases from 77.59% to 97% and sentence-level accuracy increases from 60.82% to 81.3%, BERT's sentence-level accuracy increases from 59.02% to 86.64%.

In summary, the results show that injecting city knowledge with synthesized requirements boosts the translation model significantly. While improving policy maker's user experience with higher accuracy and less clarifications, it also enhances the safety of the monitoring system potentially.

C. Performance on Online Validation

We evaluate the validation model through simulating four different testing scenarios: (I) randomly generated malicious input based on the permutation of letters and symbols; (II) all street names in Nashville; (III) real city vocabulary generated from Nashville requirements; (IV) generated float numbers with different units.

First of all, the accuracy of validation model is very high. When the uncertainty threshold is set to 0.5, i.e., all inputs cause an uncertainty higher than 0.5 will be ruled out, CitySpec gives 100% success rate against scenario I among 2,000 malicious inputs, 91.40% acceptance rate among 2,107 samples in scenario II, 92.12% acceptance rate among 596 samples in scenario III, and 94.51% acceptance rate among 2,040 samples in scenario IV. Additionally, we find that the validation function easily confuses entity with quantifier if no further guidance is offered. We look into dataset and figure out entity and quantifier are confusing to even humans without any context information. Take the requirement "In all buildings, the average concentration of Sulfur dioxide (SO2) should be no more than 0.15 mg/m3 for every day." As an example, entity is "concentration" and quantifier is "Sulfur dioxide (SO2)". If the requirement is changed to "The maximum level of the concentration of Sulfur dioxide (SO2) should be no more than 0.15 mg / m3 for every day.", then entity is "maximum level" and quantifier is "concentration" instead. In addition, terms like "occupancy of a shopping mall", "noise level at a shopping mall", and "the shopping mall of the commercial district" also introduce confusion between location, entity and quantifier, since the same token "shopping mall" can be entity, quantifier or location in certain cases.
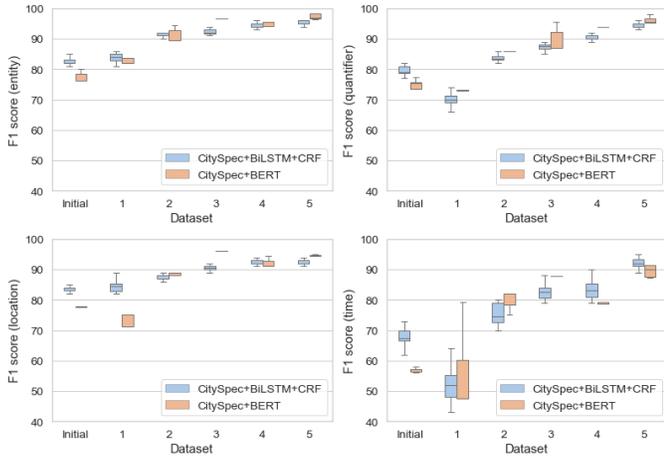
Fig. 5. F-1 scores on four keywords

The results show that the validation algorithm can effectively accept new city knowledge, prevent adversarial inputs and safeguard online learning. Therefore, CitySpec reduces unnecessary interactions between policy makers and the system and increases efficiency.

D. Adaptability to different scenarios

In this section, we analyze CitySpec's adaptability in different cities and different domains. Different cities have different regulation focuses and their city-specific vocabulary. For example, in the city of Nashville, location names like "Music Row", "Grand Ole Opry" will probably never appear in any other cities. We select four cities, Seattle, Charlottesville, Jacksonville, and Changsha, with different sizes and from different countries as case studies. We separate the requirements of each mentioned city and extract the city-wise vocabulary based on each city independently. Each of four constructed pairs consists of: vocabulary I, which is extracted from the requirements from one city only, and vocabulary II, which is extracted from the requirements from all the cities but that specific one city. Injected knowledge is measured using the number along with the ratio of how much of the unique vocabulary one city causes. We augment vocabulary II using 5 as the synthesis index and train a model on vocabulary II. As a result, the trained model is isolated from the vocabulary information from that one specific city. Afterward, we test the trained model performance on the generated requirements using vocabulary I. We pick CitySpec with Bi-LSTM + CRF and CitySpec with BERT in this scenario. We employ the validation function to validate all vocab in vocabulary I and pass the validated ones to vocabulary II. After that, we have a validated vocabulary including vocabulary II and validated vocabulary I. The deployed model is fine-tuned based on the validated vocabulary using few-shot learning.

From the results shown in Table III, we observe that (1) although CitySpec immigrates to a completely unknown city, it is still able to provide satisfying performance, e.g., 84.9% token-acc and 77.6% F-1 score in Seattle, but the sent-acc tends to be low. (2) With new knowledge injected, the

performance increases significantly, e.g., Sent-Acc for Seattle increases from 48% to 84.8% with BiLSTM+CRF, and from 46.4% to 80.7% with BERT.

We also explore CitySpec's adaptability to different topics. We choose four topics including noise control, indoor air control, security, and public access. The results also show that (1) even though CitySpec has not seen vocabulary from a totally different topic, it still gives a competitive performance; (2) online learning brings obvious improvements when adaptation is further applied.

In summary, it indicates the capability of CitySpec in both city and domain adaptation. It can also adapt to new requirements evolving overtime. Moreover, with a different set of domain-specified knowledge, CitySpec can be potentially applied to other application domains (e.g., healthcare).

E. Case Study

Due to the absence of a real city policy maker, we emulate the process of using CitySpec by taking the real-world city requirements and assuming that they are input by policy makers. Specifically, this case study shows the iteration of communication between CitySpec and the policy maker to clarify the requirements. We emulate this process 20 times with 100 requirements randomly selected from our datasets each time. The results show that the average and maximum rounds of clarification are 0.8 and 4 per requirement, respectively, due to the missing or ambiguous information. Averagely, 28.35% of requirements require clarification on location. For example, "No vendor should vend after midnight.", CitySpec asks users to clarify the time range for "after midnight" and the location defined for this requirement. Overall, CitySpec obtains an average sentence-level accuracy of 90.60% (with BERT and synthesize index = 5). The case study further proves the effectiveness of CitySpec in city requirement specification.

VI. RELATED WORK

Translation Models. Researchers have developed models to translate the natural language to machine languages in various domains, such as Bash commands [5], Seq2SQL [6], and Python codes [7]. These translation models benefit from enormous datasets. The codex was trained on a 159 GB dataset that contains over 100 billion tokens. WikiSQL, which Seq2SQL was trained on, consists of 80,654 pairs of English-SQL conversions. NL2Bash [5] was trained on approximately 10,000 pairs of natural language tasks and their corresponding bash commands. As an under-exploited domain, there is a very limited number of well-defined requirements. Therefore, existing translation models do not apply to our task. This paper develops a data synthesis-based approach to build the translation model.

Data Synthesis. Data synthesis exploits the patterns in study findings and synthesizes variations based on those patterns. Data augmentation is a simple application of data synthesis. Previous augmentation approaches wield tricks like synonym substitution [12], [13] and blended approaches [14]. In the smart city scenario, we need new data samples which fit in

TABLE III

ADAPTABILITY ON DIFFERENT CITIES IN TERMS OF TOKEN-LEVEL ACCURACY, SENT-LEVEL ACCURACY, AND OVERALL F-1 SCORE

City	Seattle			Changsha			Charlottesville			Jacksonville		
Metrics	TokenAcc	SentAcc	F-1	TokenAcc	SentAcc	F-1	TokenAcc	SentAcc	F-1	TokenAcc	SentAcc	F-1
Non-adaptive w/ BiLSTM+CRF	84.91%	48.00%	77.60%	86.61%	61.20%	84.10%	90.00%	65.62%	86.48%	77.32%	35.20%	81.54%
Adapted w/ BiLSTM+CRF	96.05%	84.80%	93.75%	95.27%	83.20%	93.57%	96.82%	89.29%	95.40%	97.35%	88.80%	96.02%
Non-adaptive w/ BERT	80.38%	46.40%	76.80%	86.10%	58.00%	83.87%	93.44%	73.21%	90.46%	90.21%	56.40%	81.60%
Adapted w/ BERT	95.10%	80.70%	90.28%	97.16%	88.40%	96.88%	97.53%	87.05%	94.31%	96.72%	83.20%	92.59%
Knowledge Injected	31 (2.34%)			23 (1.73%)			53 (4.07%)			22 (1.66%)		

TABLE IV

ADAPTABILITY ON DIFFERENT TOPICS IN TERMS OF TOKEN-LEVEL ACCURACY, SENT-LEVEL ACCURACY, AND OVERALL F-1 SCORE

Topic	Noise Control			Public Access			Indoor Air Control			Security		
Metrics	TokenAcc	SentAcc	F-1	TokenAcc	SentAcc	F-1	TokenAcc	SentAcc	F-1	TokenAcc	SentAcc	F-1
Non-adaptive w/ BiLSTM+CRF	77.82%	41.56%	77.83%	73.99%	44.07%	74.41%	81.51%	46.80%	76.22%	72.11%	28.80%	62.93%
Adapted w/ BiLSTM+CRF	95.82%	90.68%	94.46%	97.68%	74.80%	97.39%	96.58%	80.00%	87.68%	94.39%	94.37%	92.34%
Non-adaptive w/ BERT	84.15%	58.62%	81.05%	83.59%	54.17%	78.93%	78.51%	31.20%	73.88%	79.31%	45.60%	77.50%
Adapted w/ BERT	98.07%	88.31%	92.98%	97.43%	88.75%	95.75%	95.31%	74.40%	93.60%	95.41%	82.80%	93.95%
Knowledge Injected	57 (4.38%)			171 (14.37%)			92 (7.31%)			41 (3.13%)		

the smart city context. Therefore, we extract extra knowledge from smart cities and fully exploit semantic and syntactic patterns instead of applying straightforward tricks like chopping, rotating, or zooming. This paper is the first work synthesizing smart-city-specific requirements to the best of our knowledge.

Online Learning. Online machine learning mainly deals with the situation when data comes available to the machine learning model sequentially after being deployed. Similar to continuous learning, online learning aims to give model accumulated knowledge and improve model performance continuously given incoming learning samples [15], [16]. Some of the existing papers focus on developing sophisticated optimization algorithms [17] or exploiting the differences between new and old samples [18]. However, these papers do not have a mechanism to detect or prevent adversarial samples online. This paper develops a two-stage online learning process with online validation against potential malicious inputs.

VII. SUMMARY

This paper builds an intelligent assistant system, CitySpec, for requirement specification in smart cities. CitySpec bridges the gaps between city policy makers and the monitoring systems. It incorporates city knowledge into the requirement translation model and adapts to new cities and domains through online validation and learning. The evaluation results on real-world city requirement datasets show that CitySpec is able to support policy makers accurately writing and refining their requirements and outperforms the baseline approaches. In future work, we plan to have CitySpec used by real city policy makers, but this is outside the scope of this paper.

ACKNOWLEDGMENT

REFERENCES

[1] M. Ma, J. A. Stankovic, and L. Feng, "Toward formal methods for smart cities," *Computer*, vol. 54, no. 9, pp. 39–48, 2021.

[2] M. Ma, E. Bartocci, E. Lifland, J. A. Stankovic, and L. Feng, "A novel spatial–temporal specification-based monitoring system for smart cities," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 11 793–11 806, 2021.

[3] M. Ma, J. A. Stankovic, and L. Feng, "Cityresolver: a decision support system for conflict resolution in smart cities," in *2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2018, pp. 55–64.

[4] M. Ma, J. Stankovic, E. Bartocci, and L. Feng, "Predictive monitoring with logic-calibrated uncertainty for cyber-physical systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 20, no. 5s, pp. 1–25, 2021.

[5] Q. Fu, Z. Teng, J. White, and D. C. Schmidt, "A transformer-based approach for translating natural language to bash commands," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 1245–1248.

[6] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[7] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[8] F. J. Damerau, "A technique for computer detection and correction of spelling errors," *Communications of the ACM*, vol. 7, no. 3, pp. 171–176, 1964.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[10] A. X. Chang and C. D. Manning, "Sutime: A library for recognizing and normalizing time expressions." in *Lrec*, vol. 3735, 2012, p. 3740.

[11] B. E. Chapman, S. Lee, H. P. Kang, and W. W. Chapman, "Document-level classification of ct pulmonary angiography reports based on an extension of the context algorithm," *Journal of biomedical informatics*, vol. 44, no. 5, pp. 728–737, 2011.

[12] S. Kobayashi, "Contextual augmentation: Data augmentation by words with paradigmatic relations," *arXiv preprint arXiv:1805.06201*, 2018.

[13] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," *Advances in neural information processing systems*, vol. 28, pp. 649–657, 2015.

[14] J. Wei and K. Zou, "Eda: Easy data augmentation techniques for boosting performance on text classification tasks," *arXiv preprint arXiv:1901.11196*, 2019.

[15] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[16] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 1–207, 2018.

[17] E. Hazan *et al.*, "Introduction to online convex optimization," *Foundations and Trends® in Optimization*, vol. 2, no. 3-4, pp. 157–325, 2016.

[18] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.