A BERT-based Model for Semantic Consistency Checking of Automation Rules

Bernardo Breve, Gaetano Cimino, Vincenzo Deufemia, Annunziata Elefante
Department of Computer Science
University of Salerno, Italy
{bbreve, gcimino, deufemia, anelefante}@unisa.it

Abstract

Trigger-Action Platforms (TAPs) allow users to automate behaviors involving IoT devices either by programming rules from scratch or by accessing a catalog of user-defined rules. Users can search the catalog based on their interests and needs, browsing through rules that are expressed according to textual descriptions supplied by the rule's creator. However, TAPs do not perform any control over these User-defined descriptions (UDDs), which means that there is no way to ensure their suitability. This lack of control might lead to the inclusion of erroneous information, making it challenging for users to retrieve the relevant rules they need during searches. To address this issue, this paper proposes the use of a BERT-based classification model to check the semantic consistency of a rule's UDD with respect to its trigger-action components. We evaluate the proposed solution with the popular TAP, namely If-This-Then-That (IFTTT), by training the model on a dataset consisting of 9643 labeled samples. Each sample is composed of a pattern derived from the rule components, the corresponding rule's UDD, and a label expressing whether they are semantically related. The code of the software is publicly available on GitHub[1].

Index terms— Trigger-action rules, Semantic consistency checking, NLP, BERT, IoT platforms.

1 Introduction

The outburst of Internet-of-Things (IoT) leads to a new world of opportunities and challenges for programmers and end-users who use this technology [15]. In fact, these "smart" devices are spreading across houses in the form of sensors and actuators, such as cameras, lights, thermostats, locks, and so forth. Thus, an IoT device can sense the environment around it, acquiring data and sharing them over the internet with other devices, leading to an ecosystem of devices that can collaborate to generate automation [2].

To empower users in getting the most out of IoT devices by defining useful tasks such as lights turning off automatically at sunset, IoT-based applications are built, which help users define interoperability behaviors in a simple way [11]. Among the different types of platforms, the most popular are the *Trigger-Action Platforms* (TAPs), which empower users to define custom behaviors by means of conditional rules [9], consisting of a trigger component, which specifies the event whose occurrence would trigger the rule, and an action component, which describes the operation to perform in order to accomplish the behavior. In addition, most platforms allow users to specify additional information in the form of textual description, called *User-defined description* (UDD), which summarizes the behavior of the rule.

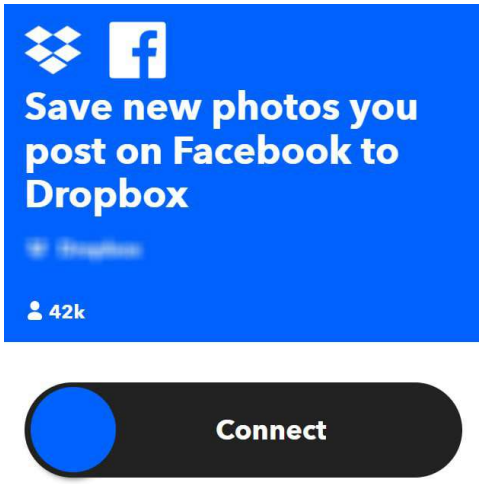

Figure 1: An applet's example with the associated UDD

If-This-Then-That (IFTTT)[2] is the most popular TAP. It was first released in 2010 and since then, it has gained a

[1] https://github.com/empathy-ws/Semantic-Consistency-Checking-of-Automation-Rules

DOI reference number: 10.18293/DMSVIVA23-008

[2] https://ifttt.com

larger number of followers. One advantage of IFTTT is the vast catalog of rules (known as *applets*) that users can access, which have been created and shared by other members of the community. In this context, UDDs become even more critical, as they help users to easily understand the behavior of applets while browsing the catalog. Figure 1 showcases an example of an IFTTT applet from the applet catalog (with the author's name blurred for privacy), which is presented based on its UDD. The UDD summarizes the applet's behavior, which in this case consists of automatically synchronizing any new photo posted by the user on Facebook to a Dropbox folder.

Unfortunately, the literature has highlighted that IoT smart environments are not risk-free with respect to security and privacy concerns [21], as devices might represent a suitable target for malicious individuals to plan cyberattacks [1]. Furthermore, users themselves can induce cyber security threats while interacting with TAPs [17]. In fact, rules created through TAPs possess inherent risks, mainly caused by the level of technical knowledge that the average TAP user has, which might not be sufficient enough to understand the severeness of an apparently innocuous rule [6]. For example, a rule such as "If I enter the gym, post a tweet with my account", might at first glance appear as an innocuous rule, however, providing a malicious individual with a routine produced by the rule, might give him/her useful insights into when planning a possible theft, aware that the house of the user will be empty. To address these concerns, the research has moved towards the definition of ad-hoc solutions for empowering users in protecting the smart environment and their privacy [3, 4, 5, 18].

The existence of fields such as UDDs also represents an important concern for users, which still has not received much attention from the literature. In fact, TAPs such as IFTTT do not perform actual control over what the authors of rules may write as a UDD, leaving them the freedom to type anything they want for describing the behavior. This may represent an issue for several reasons. First, a rule's creator may type a UDD completely unrelated not only to the behavior of the rule but also to the characteristics that a description typically should have, e.g., "You will like this rule!", making it virtually impossible for a user to find such a rule. Also, a rule with an imprecise UDD might force it to appear in the search results for other types of rules, making it even more complicated for users to identify rules that meet their needs. Finally, rules might not be understood when provided with poor UDDs, as the latter represents their showcase.

Therefore, in this paper, we address the above-mentioned concerns by proposing a classification model based on *Bidirectional Encoder Representations from Transformers* (BERT) [8] to determine the degree of semantic consistency established between the behavior of a rule and its associated UDD. To perform this comparison, we devised a methodology where the actual behavior is encapsulated within a textual pattern constructed from the rule components. The resulting pattern, along with the rule's UDD, is given as input to the classification model, which computes a semantic similarity score between the two texts. The proposed solution was evaluated on rules gathered from the IFTTT platform, using a dataset of 9643 UDD-pattern labeled pairs. The empirical analysis shows that the model effectively categorized semantic consistency, achieving an accuracy rate of approximately 92%.

The paper is organized as follows: Section 2 discusses the state of the art on semantic analysis of automation rules. Section 3 outlines the overall methodology by focusing on the dataset, the labeling process, and the model architecture. Then, in Section 4, we describe the experimental evaluation leading to the performance scores we traced back from the model. Finally, Section 5 concludes the manuscript and provides future directions for our proposal.

2 Related work

In this section, we describe the primary research efforts related to the semantic analysis of IF-THEN rules. Previous studies in the literature have focused on language-to-code methodologies, which extract executable code from rule descriptions. On the other hand, alternative studies aim to enhance the user experience by developing sophisticated graphical interfaces or by utilizing sequence-to-sequence models to automatically generate rule components, thereby simplifying the rule creation process for users.

The use of natural language to program computers could potentially increase accessibility to modern technology for inexperienced users [13]. In this regard, developing language-to-code translators could help create IF-THEN rules that cater to user needs. This may be accomplished by leveraging a semantic parser that converts natural language into executable code, thereby streamlining the process of applet customization and making it more user-friendly for a broader range of users. In [16], Quirk *et al.* designed a language-to-code approach for natural language programming. Specifically, the authors collected 114,408 applet-description pairs from the IFTTT website and used them to train semantic parser learners that could effectively interpret natural language descriptions of applet behaviors and map them to executable code. The IF-THEN statements were represented as syntactic constructs through the use of Abstract Syntax Trees (ASTs), where each node denoted a specific text construct and captured its structural and content-related details. The constructed ASTs were then fed to several classifiers, which iteratively searched for the most likely derivation, adding correct instances to the set of positive instances, and incorrect instances to

the set of negative instances. The classifiers were then retrained using the revised training data, and the process was repeated until the desired performances were achieved or a maximum number of iterations was reached. In [12], Chen *et al.* presented a neural network architecture for automatically translating natural language descriptions to IF-THEN rules. Specifically, the authors designed an attention architecture, called *Latent Attention*, that computes the importance of each word in the description for predicting rule components in a two-stage process. Yusuf *et al.* proposed *RecipeGen*, a deep learning-based approach that uses a Transformer sequence-to-sequence architecture to generate IF-THEN rules from natural language descriptions [20]. The problem is modeled as a sequence learning and generation task, which facilitates the abstraction of implicit relations between rule components. To improve the generation performance, RecipeGen relies on autoencoding pre-trained models to initialize the parameters of the encoder in the sequence-to-sequence model.

It is noteworthy that previous studies have limited their scope to interactions requiring a user's request and the system's response in the form of an interpretation. An essential aspect is to involve the user in an interactive dialogue to validate and improve their intention and create a complete and accurate rule. Concerning this matter, in [7], Corno *et al.* proposed *HeyTAP*, a conversational and semantic-powered platform that can map abstract user needs to executable IF-THEN rules. HeyTAP uses a multimodal interface to interact with the user and extract personalization intentions for different contexts. The authors conducted an exploratory experiment with 8 users to test the effectiveness of HeyTAP in guiding participants from abstract needs to actual IF-THEN rules. Results showed that HeyTAP can successfully translate abstract user needs into IF-THEN rules that can be executed by contemporary TAPs. While previous semantic parsers, as presented, perform both text parsing and comprehension in a single step, Yao *et al.* proposed an approach relying on a Hierarchical Reinforcement Learning framework to translate natural language descriptions into IFTTT applets [19]. This approach introduces an interactive element to semantic analysis, where an agent is trained with a hierarchical policy to maximize the parsing accuracy while minimizing the number of questions asked to the user. Finally, Huang *et al.* conducted a thorough analysis of the potential implications of incorporating natural language interfaces for assisting users in the customization and automation of their personal devices [10]. In particular, the authors introduced *InstructableCrowd*, a crowd-powered system that enables users to program their devices via a natural language interface. The system is based on two key design decisions: i) creating simple programs that are easy to use and ii) employing human crowd workers to operate the natural language interface instead of using automated systems. The system is oriented around relatively simple IF-THEN rules and contains more than one sensor/effector. The authors argue that InstructableCrowd addresses the main problems with device customization and automation, and could provide a new way to program devices in the future.

With respect to the approaches that focus on analyzing natural language descriptions to generate executable rules [7, 12, 16, 19, 20], or that investigate how to interact with users in order to improve the rule definition process [10], we address a different problem since we focus on checking the semantic consistency of a UDD against the actual rule behavior before its dissemination.

3 Methodology

In this section, we outline the methodology used to develop a model that evaluates the semantic consistency between the trigger-action components of an IFTTT applet and the natural language description provided by the creator. Specifically, we describe the dataset employed during the experimental evaluations and the technical details for implementing the semantic consistency evaluation model.

3.1 IFTTT Applet Dataset

In our study, we utilized the dataset proposed by *Mi et al.* [14], which consists of a collection of IFTTT applets obtained from crawling the IFTTT.com website. This dataset includes crucial information such as a title (*Title*), a description explaining the applet behavior (*Desc*), the event triggering the applet (*TriggerTitle*) defined through a specific channel (*TriggerChannelTitle*), the action to be performed (*ActionTitle*) selected from the corresponding channel (*ActionChannelTitle*), and the name of the applet creator (*Creator Name*). We employed the information generated by IFTTT to design a new pattern for *synthesizing* UDDs, ensuring a coherent and accurate representation. Then, we performed *dataset labeling*, a critical process that involved assigning suitable labels to the synthesized dataset to facilitate efficient categorization, organization, and analysis of the data.

3.1.1 Synthesizing a UDD from the components of an applet

To evaluate how consistent a UDD is with an applet's actual behavior, we designed a pattern that acts as a natural language description by leveraging applet key components, including trigger, trigger channel, action, and action channel. In particular, the pattern utilized for generating the synthesized UDD is as follows:

> ***IF*** *TriggerTitle (TriggerChannelTitle)* ***THEN*** *ActionTitle (ActionChannelTitle)*

This standardized structure serves as a concise and comprehensive means of depicting the essential components and events associated with a specific applet. To further illustrate, we provide an example employing an IFTTT applet consisting of the following components:

- ***TriggerTitle***: "Any new SMS received"
- ***TriggerChannelTitle***: "Android SMS"
- ***ActionTitle***: "Send me an email"
- ***ActionChannelTitle***: "Email"

The pattern generated for this applet is as follows:

> ***IF*** *Any new SMS received (Android SMS)* ***THEN*** *Send me an email (Email)*

This pattern provides a clear and concise representation of the applet's components and their corresponding values, enabling a comprehensive understanding of its intended behavior. This statement remains true even after reading the original description:

> *When a text message arrives, forwards it to your email.*

3.1.2 Dataset labeling

Another crucial aspect is the construction of a set for the training phase of the proposed model. In this regard, we randomly selected a subset of applets and labeled them over a period of three weeks. The labels were selected based on the correlation between the UDD and the description synthesized by the pattern presented in the previous section. In particular, the UDD-pattern pairs were labeled according to the following similarity label values:

- ***contradiction***: denotes inconsistency between the UDD and the synthesized pattern.
- ***entailment***: denotes consistency between the UDD and the synthesized pattern.

The labeling of the pairs was determined based on the following conditions: If a UDD accurately depicted both the trigger and action components of a rule, it was assigned the label entailment. Otherwise, the UDD-pattern pair was labeled as contradiction. Specifically, we applied the majority method for manually labeling the pairs of the considered dataset. The first, second, and fourth authors were in charge of manually labeling the pairs, with the third author intervening in cases where there was no agreement. With this strategy, we obtained a dataset containing 10,543 labeled pairs, where 6,540 belong to class entailment and 4,003 to class contradiction. This careful labeling of pairs ensures that our model is trained on a diverse and representative set of data, enabling reliable evaluations and accurate assessments of semantic consistency between applet patterns and descriptions.

The resulting dataset was used to train and evaluate the BERT-based classification model adopted to perform the semantic consistency checking task.

3.2 The Proposed BERT-based Model

The architecture of the proposed model for classifying the semantic consistency of an applet's UDD with respect to the corresponding pattern is depicted in Figure 2. The model comprises several interconnected components working together to achieve our goal.

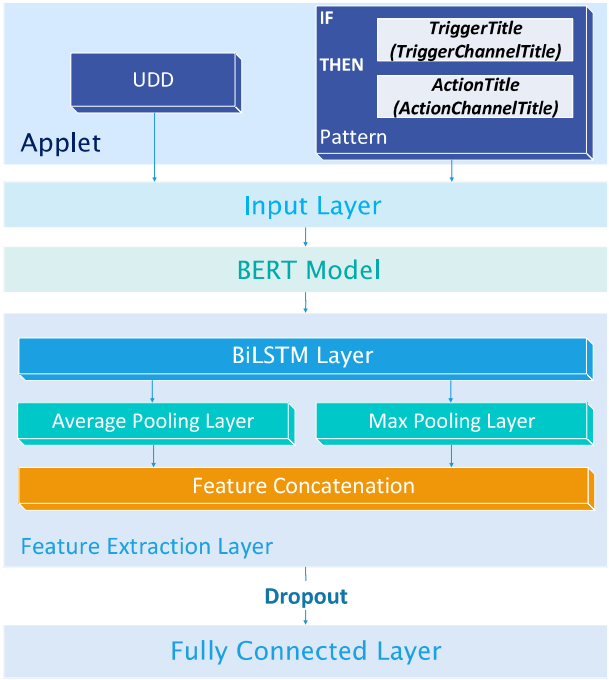

Figure 2: The architecture of the BERT-based model

The first component is an *Input Layer* that takes the UDD-pattern pairs from the dataset and encodes them into numerical representations, also known as dense-type vectors. Once the texts are transformed into dense vectors, they are passed to a state-of-the-art language model that has been pre-trained on a large corpus of text, namely BERT. The latter consists of multiple Transformer Encoder Layers that generate contextual representations of each word in

the input sequence by leveraging the self-attention mechanism. Each layer outputs a series of dense vectors that capture different levels of syntactic and semantic information. The next step involves passing the sequence obtained from the BERT model as input to a *Feature Extraction Layer*. This layer includes a *Bidirectional Long Short-Term Memory* (BiLSTM) *Layer*, which is a variant of the traditional LSTM Layer that is designed to store the both past and future context of a sequence. In this case, the BiLSTM Layer consists of 64 LSTM cells that are connected in a chain. The output of the BiLSTM Layer is a sequence of vectors, where each vector represents the hidden state of the LSTM cell at a given time step. These hidden states are then concatenated to form a representation of the input sequence that captures global features and dependencies. The output from the BiLSTM Layer has then proceeded through two pooling layers, i.e., an *Average Pooling Layer* and a *Max Pooling Layer*, which reduce the dimensionality of the input data by aggregating information across the sequence. In particular, the first type of pooling calculates the average value of each feature throughout the sequence, capturing the overall representation and distribution of the features. In this way, it can help mitigate the impact of outliers or extreme values in the sequence. On the other hand, the second one selects the maximum value from each dimension of the vectors, capturing salient and important feature values. Thus, it can help highlight the most relevant information and discard less important details, which can be beneficial for identifying key elements or detecting specific patterns. The resulting vectors are concatenated into a single one by a *Feature Concatenation* module, which produces a compact representation of the input texts. Specifically, average pooling provides a global representation of the sequence, while max pooling focuses on the most significant features. Therefore, concatenating the results of both pooling operations into one vector allows the model to have a comprehensive representation that captures both the overall context and important local details. Before feeding the concatenated data into the *Fully Connected Layer*, a *Dropout* operation is applied. This randomly drops out a fraction of the input features, preventing the model from relying too heavily on any single feature and mitigating the risk of overfitting. Finally, the Fully Connected Layer exploits the extracted features to evaluate the semantic consistency of the applet descriptions with respect to the corresponding patterns. In particular, it uses the vector obtained from the previous layer as input and applies a series of linear transformations to compute the final classification output of the model.

4 Experimental Evaluation

In this section, we present an analysis of the performances of the implemented model. Specifically, we provide details on the experimental setup, the adopted metrics, and the results obtained from the experiments.

4.1 Evaluation Setup

We trained the BERT-based model through a two-step process. Initially, we froze all the pre-trained layers and performed a training process solely targeting the top layers of the model. This enabled feature extraction by exploiting the representations of the pre-trained model. After the feature extraction process, we performed an additional fine-tuning step. This involved unfreezing the BERT model and retraining the entire architecture using a considerably low learning rate. The purpose of this step was to progressively adapt the pre-trained features to the new data, significantly enhancing the performances of the model.

To pre-train and tune the proposed model for our semantic consistency checking task, we used the Python libraries `Keras` and `TensorFlow`. Among the range of pre-trained BERT models currently available, we used the "bert-base-uncased" variant in our study. This model corresponds to the "base" version, featuring 12 transformer blocks, 768 hidden units, and 12 self-attention heads. Additionally, it is designed to account for lowercase letters. The training set included 6006 entailment pairs and 3637 contradiction pairs, totaling 9643 samples. Furthermore, in order to determine the best hyperparameter configuration, we employed a validation set consisting of 150 entailment pairs and 150 contradiction pairs, yielding as best values: 4 epochs, 32 as batch size, 1e-5 as epsilon, and 70 as a maximum text length. Finally, we evaluated the model's performances using a test set comprising 384 entailment pairs and 216 contradiction pairs.

4.2 Evaluation Metrics

The evaluation of the proposed model's performances is based on multiple metrics, including Accuracy, Precision, Recall, and F1-score. Specifically, the assessment metrics are derived from the values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). Accordingly, the evaluation metrics are formulated as follows:

- **Accuracy** is a measure of the overall correctness of a model's predictions, expressed as the ratio of the number of correctly classified instances to the total number of instances evaluated: $\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}}$
- **Precision** is a measure of the proportion of true positive instances among all instances that the model identified as positive: $\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}}$

- **Recall** is a measure of the proportion of true positive instances among all actual positive instances: $\text{Recall} = \frac{\text{TP}}{\text{TP+FN}}$

- **F1-score** is the harmonic mean of Precision and Recall: $\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}}$

4.3 Results and Discussion

Figure 3 shows the confusion matrix obtained from the classification results on the test set, whereas Table 1 reports the resulting values of Accuracy, Precision, Recall, F1-score, and the average of the per-class metrics.

Figure 3: Confusion Matrix of the BERT-based model

We can observe that the model exhibits satisfactory performances in discriminating between entailment and contradiction UDD-pattern pairs, as highlighted by the Accuracy value of 92%. Upon analyzing the values for individual classes, it is feasible to note that the model primarily identifies entailment pairs, achieving Recall and F1-score values of 97% and 94%, respectively. Nevertheless, it is worth noting that the model tends to produce the entailment class more frequently than the contradiction class, leading to a lower Recall value for the contradiction class (83%). In particular, it wrongly classifies a contradiction pair as entailment 36 times, influencing the Precision value of this class (91%). An in-depth manual analysis of classification results reveals that this scenario is due to the structure of UDDs, which despite providing high knowledge about applet behaviors, may not include enough detail to understand them completely. As an example, consider the following pair:

> **Pattern:** *If a new photo post by you with a hashtag on Facebook then upload the file from the URL on Google Drive*
> **UDD:** *Crowdsource wedding photos from Facebook with a hashtag*

This pair is labeled as contradiction because, while presenting some correct information about the applet's behavior, the description does not perfectly explain the action to be performed. In this case, the model misclassifies the pair as entailment, probably due to the fact that it focuses significant attention on the actual relevant part of the text. In general, we can argue that the model might misjudge a pair when a user customizes a description based on how s/he will use the applet rather than specifying the components of the applet. On the other hand, we can observe that the model shows high reliability when classifying a pair with the contradiction class, as proved by the few cases where an entailment pair is classified as contradiction, i.e., 13 times, resulting in a high Precision value for this class (93%).

Table 1: Classification performances on the test set

Metric	Contradiction	Entailment	Avg
Precision (%)	93	91	92
Recall (%)	83	97	90
F1-score (%)	88	94	91
Accuracy (%)			92

5 Conclusion and Future Work

TAPs have enabled all types of users to easily define complex automation related to IoT devices by means of simple conditional rules. Such rules can then be described through UDDs freely typed by users without there being an effective check by the platforms on the correctness of what is written. Thus, in this paper, we addressed this issue by proposing a BERT-based classification model for evaluating the semantic consistency between the UDD of a rule and its actual behavior, the latter inferred by analyzing the trigger and action rule components. Our experimental evaluation over a case study on the IFTTT platform, considering a dataset of 9643 manually labeled UDD-pattern pairs, revealed an overall accuracy rate of 92%, indicating high reliability of the model in discriminating a compliant UDD with respect to an unrelated one.

In the future, we would like to consider the adoption of Large Language Models (LLMs) to generate, from the rules components, ad-hoc descriptions which might be presented to the user as a suggestion, perhaps in place of a less

suitable UDD. Furthermore, we might consider introducing additional classification outputs for the model, indicating a description that, although it contains some correct information, it is not sufficiently detailed to make clear the behavior of the rule from the perspective of both the trigger and action components.

Acknowledgements

This work has been supported by the Italian Ministry of University and Research (MUR) under grant PRIN 2017 "EMPATHY: Empowering People in deAling with internet of THings ecosYstems" (Progetti di Rilevante Interesse Nazionale – Bando 2017, Grant 2017MX9T7H).

We thank Francesca Cerruto for supporting the research goals of this work.

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou. Understanding the Mirai Botnet. In *Proceedings of the 26th USENIX Conference on Security Symposium*, SEC'17, page 1093–1110, USA, 2017. USENIX Association.

[2] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] B. Breve, G. Cimino, and V. Deufemia. Towards explainable security for ECA rules. In *Proceedings of the 3rd International Workshop on Empowering End-Users in Dealing with Internet of Things Ecosystems*, EMPATHY'22, 2022.

[4] B. Breve, G. Cimino, and V. Deufemia. Identifying security and privacy violation rules in trigger-action IoT platforms with NLP models. *IEEE Internet of Things Journal*, 10(6):5607–5622, 2023.

[5] B. Breve, G. Desolda, V. Deufemia, F. Greco, and M. Matera. An end-user development approach to secure smart environments. In *Proceedings of 8th International Symposium on End-User Development*, IS-EUD'21, pages 36–52. Springer, 2021.

[6] C. Cobb, M. Surbatovich, A. Kawakami, M. Sharif, L. Bauer, A. Das, and L. Jia. How risky are real users' IFTTT applets? In *Proceedings of the Sixteenth USENIX Conference on Usable Privacy and Security*, pages 505–529, 2020.

[7] F. Corno, L. De Russis, and A. Monge Roffarello. Hey-TAP: Bridging the gaps between users' needs and technology in IF-THEN rules via conversation. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI'20, pages 1–9, 2020.

[8] 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.

[9] G. Ghiani, M. Manca, F. Paternò, and C. Santoro. Personalization of context-dependent applications through trigger-action rules. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 24(2):1–33, 2017.

[10] T.-H. Huang, A. Azaria, O. J. Romero, and J. P. Bigham. Instructablecrowd: Creating if-then rules for smartphones via conversations with the crowd. *arXiv preprint arXiv:1909.05725*, 2019.

[11] A. Krishna, M. Le Pallec, R. Mateescu, and G. Salaün. Design and deployment of expressive and correct web of things applications. *ACM Trans. Internet Technol.*, 3(1):1–30, 2021.

[12] C. Liu, X. Chen, E. C. Shin, M. Chen, and D. Song. Latent attention for if-then program synthesis. *Advances in Neural Information Processing Systems*, 29, 2016.

[13] B. Manaris. Natural language processing: A human-computer interaction perspective. In *Advances in Computers*, volume 47, pages 1–66. Elsevier, 1998.

[14] X. Mi, F. Qian, Y. Zhang, and X. Wang. An empirical characterization of ifttt: ecosystem, usage, and performance. In *Proceedings of the 2017 Internet Measurement Conference*, pages 398–404, 2017.

[15] S. C. Mukhopadhyay and N. K. Suryadevara. *Internet of things: Challenges and opportunities*. Springer, 2014.

[16] C. Quirk, R. Mooney, and M. Galley. Language to code: Learning semantic parsers for if-this-then-that recipes. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 878–888, 2015.

[17] Q. Wang, P. Datta, W. Yang, S. Liu, A. Bates, and C. A. Gunter. Charting the attack surface of trigger-action IoT platforms. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, page 1439–1453, New York, NY, USA, 2019. Association for Computing Machinery.

[18] D. Xiao, Q. Wang, M. Cai, Z. Zhu, and W. Zhao. A3ID: an automatic and interpretable implicit interference detection method for smart home via knowledge graph. *IEEE Internet of Things Journal*, 7(3):2197–2211, 2019.

[19] Z. Yao, X. Li, J. Gao, B. Sadler, and H. Sun. Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019.

[20] I. N. B. Yusuf, L. Jiang, and D. Lo. Accurate generation of trigger-action programs with domain-adapted sequence-to-sequence learning. In *Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension*, ICPC '22, page 99–110, New York, NY, USA, 2022. Association for Computing Machinery.

[21] E. Zeng, S. Mare, and F. Roesner. End user security and privacy concerns with smart homes. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 65–80, Santa Clara, CA, July 2017. USENIX Association.