Journal of Visual Language and Computing

journal homepage: www.ksiresearch.org/jvlc

Unleashing the Power of NLP Models for Semantic Consistency Checking of Automation Rules

Bernardo Breve[a], Gaetano Cimino[a], Vincenzo Deufemia[a] and Annunziata Elefante[a,*]

[a]University of Salerno, via Giovanni Paolo II, Fisciano (SA), 84084, Italy

ARTICLE INFO

Article History:
Submitted 4.30.2023
Revised 7.31.2023
Accepted 8.1.2023

Keywords:
Trigger-action rules
Semantic consistency checking
NLP
IoT platforms

ABSTRACT

Trigger-Action Platforms (TAPs) empower users to automate tasks involving smart devices, allowing them to either create rules from scratch or access a catalog of existing user-defined rules. Users can explore the catalog and find rules based on their interests, relying on the so-called User-defined descriptions (UDDs) provided by the rules' creators. However, TAPs lack a mechanism to verify or regulate these descriptions, resulting in potential inaccuracies or errors. This poses challenges for users when seeking relevant rules, as descriptions may present misleading or irrelevant information.

In this paper, we propose a novel approach to semantically validate the consistency between a rule's UDD and its trigger-action components. To accomplish this objective, we used rules derived from a widely used TAP, i.e., If-This-Then-That (IFTTT). From the automation rules, we constructed a dataset of 20,000 samples, and we assigned them labels representing four distinct classes of semantic consistency. For two of these classes, we leveraged the capabilities of a Large Language Model (LLM) to edit the user descriptions, significantly reducing manual effort while ensuring coherent samples. In order to evaluate the semantic consistency, we employed three NLP-based classification models, fine-tuned on the dataset we created. This allowed us to assess the effectiveness of our proposed approach. Among the models, the BERT-based model demonstrated superior performance, achieving an accuracy value of 99%.

1. Introduction

The Internet of Things (IoT) has revolutionized various industries and aspects of daily life by connecting physical devices and enabling data exchange through sensors and network connectivity [20]. Intelligent IoT systems and devices enable the automation of tasks and efficient data management, leading to the emergence of "smart devices" that enhance user experiences. Trigger-Action Platforms (TAPs) [13, 34] are crucial pieces of software in IoT systems, as they allow users to create automation rules that trigger specific actions based on conditions, such as turning on the light automatically at a certain time. TAPs are particularly valuable in End-User Development (EUD) [19, 26], as they empower users to define their automation tasks without the need for extensive programming knowledge in a very simple and intuitive way. This user-friendly approach opens up endless possibilities for customization and tailoring automation to individual needs and preferences.

Each rule is composed of a trigger component, defining the event that activates the rule, and an action component, detailing the operation to be executed to achieve the desired behavior. Additionally, TAPs often allow users to provide rule-specific information in the form of a textual description, known as the *User-defined description* (UDD), which succinctly summarizes the rule's behavior.

The If-This-Then-That (IFTTT) [1] platform serves as the primary and most widely used TAP in the market. Since its inception in 2010, the platform has garnered a substantial

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).

*Corresponding author
bbreve@unisa.it (B. Breve); gcimino@unisa.it (G. Cimino); deufemia@unisa.it (V. Deufemia); anelefante@unisa.it (A. Elefante)
ORCID(s): 0000-0002-3898-7512 (B. Breve); 0000-0001-8061-7104 (G. Cimino); 0000-0002-6711-3590 (V. Deufemia); 0009-0001-7141-6105 (A. Elefante)

[1] https://ifttt.com

DOI reference number: 10.18293/JVLC2023-N2-035

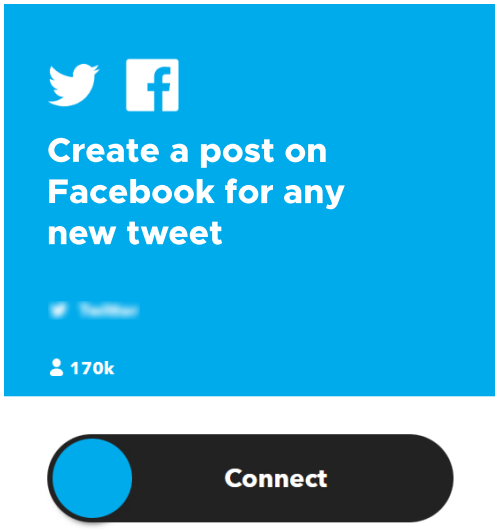

Figure 1: An rule's example with the associated UDD

and ever-growing community of followers. A notable advantage of IFTTT is its extensive catalog of rules, known as *applets*, shared by community members. In this context, UDDs play a crucial role in helping users comprehend rule behaviors while browsing the catalog. Figure 1 presents an example of an IFTTT rule from the catalog (with the author's name blurred for privacy), with the rule's behavior summarized through its UDD. In this specific instance, the rule automates the synchronization of any new tweet published by a user on Twitter to his/her personal Facebook Page.

Sadly, the utilization of TAPs and automation rules introduces security and privacy risks [10, 16, 38], as they may grant access to sensitive data and be misinterpreted in their behavior. The presence of IoT devices also poses potential risks, as they could be exploited by malicious individuals for cyber attacks [1, 25]. Additionally, users' interactions with TAPs can inadvertently introduce cybersecurity threats [33]. The creation of rules through TAPs may carry inherent risk, particularly due to the average user's level of technical knowledge, which may not be sufficient to fully comprehend the potential consequences of seemingly innocuous rules [11, 30]. For instance, a rule like "If the last family member leaves the house, then turn off the lights" could inadvertently disclose valuable information to malicious individuals, providing insights into when the user's home will be empty. To address these challenges, researchers have proposed tailor-made solutions to safeguard users' privacy and security within intelligent environments [3, 9, 35].

The existence of fields like UDDs also raises significant concerns for users [2, 7], yet this aspect has received limited attention in the literature. TAPs such as IFTTT lack active control over the content authors may input as UDDs, granting them the freedom to write anything to describe the behavior of their rules. This unrestricted approach may give rise to several issues. Firstly, rule creators may enter UDDs that are completely unrelated not only to the rule's behavior but also fail to conform to the typical characteristics of a description, such as "10 Things You Need To Know!". Consequently, such rules become virtually impossible for users to discover. Secondly, rules with imprecise UDDs might surface in search results for other types of rules, making it even more challenging for users to find rules that suit their requirements accurately. Moreover, poor UDDs may lead to a lack of understanding of the rules' intended behavior, as the UDD serves as a showcase for the rule's purpose. Finally, TAPs with shared rule catalogs can potentially expose users to risks if malicious authors hide harmful behaviors behind misleading descriptions. For instance, consider the rule's UDD in Figure 1, but assume that its trigger and action components instead are "If anyone in your area publishes a new tweet" and "then create a post on Facebook", respectively. The combination of these components could potentially result in the malicious posting of embarrassing and unwanted texts. As a consequence, such information may automatically be published on Facebook and shared with an even wider audience without the user's consent or awareness, unlike what the UDD might suggest. Since both the UDD and the actual trigger-action components involve the same services, i.e., *Twitter* and *Facebook*, users might be deceived into activating such a rule, unaware of the potential harm. These concerns emphasize the need for attention and possible solutions in this domain.

To mitigate this risk is crucial to preserve the semantic consistency between a rule's behavior and its UDD. This should be done by means of approaches that analyze UDDs to identify potential misalignments, safeguarding users from potential threats that may arise due to deceptive or misleading rule descriptions. Furthermore, maintaining semantic consistency between a rule's behavior and its UDD is important also for those approaches relying on the analysis of UDDs to identify potential user privacy or security-related harm caused by rules [5, 15, 31].

In response to the mentioned concerns, this paper proposes a novel approach that addresses the identified issues effectively. In previous work [6], we proposed a Bidirectional Encoder Representations from Transformers (BERT)-based model evaluating the semantic consistency between UDDs and the trigger-action components of rules according to two consistency classes, i.e., either complete consistency between a UDD and its trigger-action component, or complete inconsistency. In this paper, we further extend our methodology by considering two additional classes of semantic consistency, i.e., trigger-side inconsistent only and action-side inconsistent only. Furthermore, we also compared the BERT-based model [14] with two other transformer-based NLP ones: the Generative Pre-trained Transformer 2 (GPT-2) model [28], which is a decoder-only model, and the Text-To-Text Transfer Transformer (T5) model [29], which combines both encoder and decoder components.

To obtain the samples to be considered for the training following the new distribution of classes, we constructed a dataset encompassing all conceivable scenarios related to the generation and sharing of UDDs associated with automation rules. Leveraging a Large Language Model (LLM) [24],

we significantly reduce manual efforts while ensuring dataset heterogeneity. The resulting dataset contains 20,000 samples, where the actual behavior of each rule is represented by a textual pattern derived from its components. These patterns, along with the rule's UDDs, serve as inputs for the classification models, which calculate a semantic similarity score between the two texts. The results demonstrated the effectiveness of these models in categorizing semantic consistency, achieving an overall accuracy rate of approximately 99%, 97%, and 91% respectively.

The paper is organized as follows: Section 2 discusses the state of the art on semantic analysis of automation rules. Section 3 presents an overview of the overall methodology, detailing the model's general architecture and outlining the dataset construction process. Moving on to Section 4, we describe the process of generating the dataset, encompassing all possible types of UDDs that a user might define. For the task of checking the semantic consistency between a rule's UDD and its trigger-action components, Section 5 provides an in-depth explanation of the NLP classification models employed. In Section 6, we present the results of the experimental evaluation, assessing the effectiveness of the classification models. Finally, Section 7 concludes the manuscript and provides future directions for our proposal.

2. Related Work

This section presents an overview of the main research endeavors in the world of semantic analysis of trigger-action rules. Previous studies in the literature have primarily concentrated on language-to-code approaches, extracting executable code from rule descriptions. Alternatively, there have been efforts to improve user experience by developing advanced graphical interfaces or employing sequence-to-sequence models for the automatic generation of rule components, streamlining the rule creation process for users.

Utilizing natural language to program computers has the potential to enhance accessibility to modern technology, especially for inexperienced users [22]. One approach to achieve this is through the development of language-to-code translators, which aid in creating trigger-action rules tailored to user needs. By employing a semantic parser, natural language descriptions can be converted into executable code, streamlining the process of rule customization and making it more user-friendly for a broader audience. In [27], Quirk *et al.* designed a language-to-code approach for natural language programming. They collected a significant number of rule-description pairs from the IFTTT website and used them to train semantic parser learners capable of effectively interpreting natural language descriptions and mapping them to executable code. The IF-THEN statements were represented using Abstract Syntax Trees (ASTs), with each node denoting a specific text construct and capturing its structural and content-related details. The constructed ASTs were then fed to several classifiers, which iteratively searched for the most likely derivation, refining the training data to achieve desired performance. Another study [21] by Chen *et al.* proposed a neural network architecture for automatically translating natural language descriptions into IF-THEN rules. They introduced an attention mechanism called *Latent Attention*, which computed the importance of each word in the description to predict rule components in a two-stage process. Additionally, Yusuf *et al.* presented *RecipeGen*, a deep learning-based approach that utilizes a Transformer sequence-to-sequence architecture to generate IF-THEN rules from natural language descriptions [37]. This model treated the problem as a sequence learning and generation task, effectively capturing implicit relations between rule components. To enhance generation performance, RecipeGen relied on autoencoding pre-trained models to initialize the encoder's parameters in the sequence-to-sequence model.

Prior studies have mainly focused on interactions that involve a user's request and the system's response in the form of interpretation. However, it is essential to engage the user in an interactive dialogue to validate and refine their intentions, leading to the creation of complete and accurate rules. Addressing this aspect, Corno *et al.* proposed *HeyTAP* [12], a conversational and semantic-powered platform that can map abstract user needs to executable IF-THEN rules. HeyTAP utilizes a multimodal interface to interact with the user and extract personalization intentions for various contexts. An exploratory experiment involving 8 users demonstrated HeyTAP's effectiveness in guiding participants from abstract needs to concrete IF-THEN rules, which can be executed by contemporary TAPs. In contrast, Yao *et al.* [36] presented an approach that introduced an interactive element to semantic analysis. They relied on a Hierarchical Reinforcement Learning framework to translate natural language descriptions into IFTTT rules. The approach involved training an agent with a hierarchical policy to maximize parsing accuracy while minimizing the number of questions asked to the user, making the process more efficient and user-friendly. Additionally, Huang *et al.* [18] conducted an in-depth analysis of the potential implications of incorporating natural language interfaces to assist users in customizing and automating their personal devices. They introduced *InstructableCrowd*, a crowd-powered system enabling users to program their devices via a natural language interface. The system focuses on creating simple programs that are easy to use and employs human crowd workers to operate the natural language interface. By incorporating more than one sensor/effector, InstructableCrowd addresses key problems with device customization and automation, offering a promising approach for programming devices in the future.

Unlike the approaches that concentrate on generating executable rules from natural language descriptions [12, 21, 27, 36, 37], or aim to enhance the rule definition process through user interactions [18], we address a different problem. We focus on checking the semantic consistency of a UDD against the actual rule behavior before its dissemination. This ensures that the UDD accurately represents the intended behavior of the rule and reduces the risk of misleading or deceptive rule descriptions.

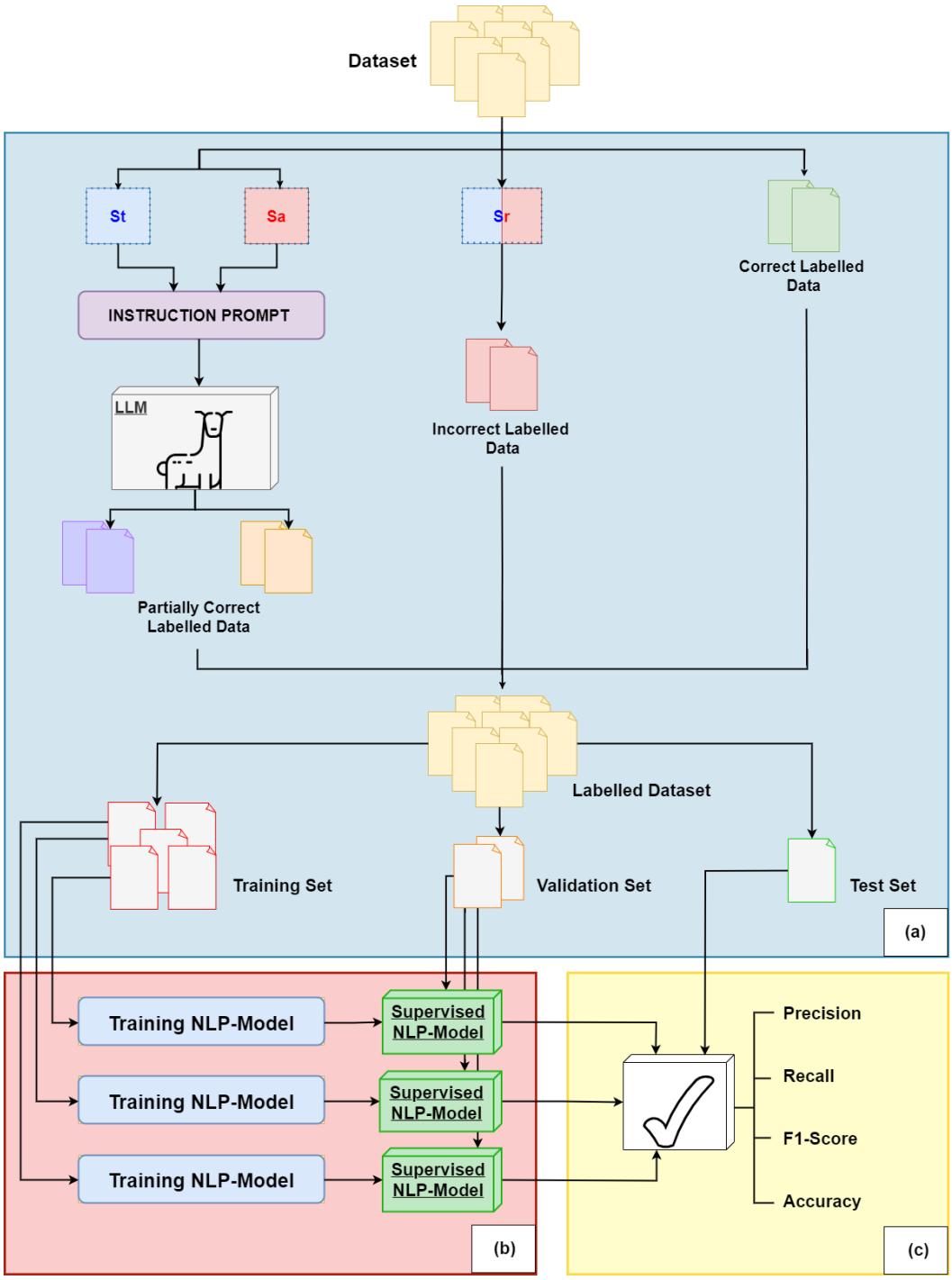

Figure 2: Proposed process for constructing the dataset and evaluating NLP-based models for checking the semantic consistency of a rule's UDD with respect to its trigger-action components. (a) Dataset construction. (b) Models training. (c) Models testing.

3. Methodology

In this section, we outline the approaches undertaken to build a comprehensive dataset encompassing UDD samples and to identify the most suitable NLP classification model for our specific objectives. In particular, Figure 2 illustrates the step-by-step process leading to the creation of the dataset and the establishment of effective supervised models to accurately examine the semantic relationships between UDDs and the synthesized patterns of their trigger-action components.

This process involves three main phases:

a) *Building and Labeling the Dataset*: The primary objective of this step is to prepare the labeled dataset required for training the NLP classification models. However, before initiating this procedure, it is essential to establish a pattern that effectively synthesizes

the behavioral aspects of a rule based on its trigger-action components. This pattern will serve as a reference for analyzing the corresponding UDD.

Furthermore, it is imperative to define the possible consistency classes for the UDD-pattern pairs and their associated labels. These classes categorize the UDDs into different types based on their alignment with the synthesized patterns. To achieve this, we first worked with the original dataset and partitioned it into four distinct samples. Each sample corresponds to different types of UDDs that a user might define for their rule, resulting in a diverse representation of descriptions across various consistency scenarios. By doing so, we ensure that each sample represents a consistent class, simplifying the labeling process significantly.

To account for various real-world scenarios, it was crucial to introduce shuffling mechanisms during the construction of three of these samples. These shuffling mechanisms (depicted as "*Sr*", "*St*", and "*Sa*" in Figure 2) introduce negative and partially negative samples of UDDs, considering that users might define descriptions that are inconsistent with the trigger component, the action component, or both. As a result, the shuffling mechanisms randomly modify one or both components of the rules in these samples, allowing us to incorporate samples of inconsistent and partially inconsistent UDDs.

Finally, all the samples are consolidated to create a labeled dataset that includes all types of samples required for training the NLP classification models. The dataset now comprises a large set of rules labeled according to their respective consistency classes, providing a solid foundation for the subsequent model training and evaluation stages.

b) *Training NLP Classification Models*: This stage focuses on training classification models using the labeled UDD-pattern pair dataset, referred to as the training set. The features used for training the models encompass the textual representation of the UDD, along with the corresponding synthesized pattern. By leveraging NLP techniques, we can extract crucial semantic information from these components, which the classification models can utilize to discern and distinguish among various consistency classes.

To ensure the effectiveness and accuracy of the classification models, we carried out a meticulous phase of dataset construction. As a result, we achieved a balanced training set, wherein each consistency class is evenly represented by 5,000 samples. This balanced distribution eradicates the issue of some classes being more frequent than others, preventing potential bias and ensuring that the models are equally well-trained on all consistency scenarios.

c) *Testing NLP Classification Models*: In this phase, our main objective is to thoroughly evaluate the performance of the NLP classification models. This evaluation is accomplished by providing the classification models with a carefully selected set of labeled UDD-pattern pairs, which serves as the input for testing their capabilities in determining semantic consistency.

To measure the effectiveness and accuracy of the models, we employ well-known evaluation metrics, i.e., Precision, Recall, F1-score, and Accuracy. By using these widely recognized evaluation metrics, we can effectively assess and compare the performance of the NLP classification models in the context of semantic consistency checking. These metrics provide valuable insights into the models' strengths and weaknesses, allowing us to make informed decisions about their suitability for real-world applications.

In the following sections, we provide a comprehensive description of the steps involved in constructing the labeled dataset tailored to serve our specific objectives. This dataset forms a crucial foundation for our research, facilitating the training and evaluation of the NLP classification models employed in our task.

4. Dataset Construction

As mentioned above, this section aims to outline the process of dataset construction and labeling, which serves as the foundation for training classification models dedicated to assessing the semantic consistency between the trigger-action components of an IFTTT rule and the accompanying natural language description. Our focus is on presenting both the starting dataset employed during analysis and the technical mechanisms involved in generating a new dataset specifically tailored for semantic consistency evaluation.

4.1. IFTTT Rule Dataset

In our study, we utilized the dataset proposed by *Mi et al.* [23], which provides a collection of IFTTT rules obtained from crawling the IFTTT.com website. The dataset contains important information such as the rule's title (*Title*), a description explaining the rule behavior (*Desc*), the event that triggers the rule (*TriggerTitle*) defined through a specific channel (*TriggerChannelTitle*), the action to be performed (*ActionTitle*) selected from the corresponding channel (*ActionChannelTitle*), and the name of the rule creator (*Creator Name*).

Exploiting the valuable information provided by IFTTT, we embarked on a novel approach for building a new dataset by devising a specialized pattern for *synthesizing UDDs*.

These patterns were meticulously designed to ensure a coherent and precise representation of a rule's behavior, capturing essential details about its trigger and action components presented in the original dataset. These structured patterns played a pivotal role in our research, serving two significant tasks.

The first task centered around the core objective of our study: the semantic consistency checking task. By employing the synthesized patterns, we were able to assess the se-

mantic alignment between a rule's trigger-action components and the corresponding UDD.

The second task involved harnessing the potential of Large Language Models (LLMs). By generating new random patterns using our structured approach, we presented these patterns as input to the LLM. This enabled us to generate samples of erroneous descriptions that users might write, encompassing incorrect triggers, actions, or even both components. This allowed for a comprehensive examination of potential user errors, broadening the scope of our analysis beyond just consistent samples.

In the final stage, we carried out *dataset labeling* by categorizing each UDD-pattern pair into its defined consistency class. This critical process significantly enhanced the efficiency of data organization and analysis. By accurately labeling the pairs, we ensured that our dataset encompassed a diverse representation of consistency scenarios, including complete consistency, complete inconsistency, and partial consistency of a UDD.

4.2. Synthesizing a UDD from the components of a rule

In our earlier study, we devised a structure that functions as a natural language description to evaluate the coherence between a UDD and the real behavior of a rule. This structure incorporates essential rule elements such as *trigger*, *trigger channel*, *action*, and *action channel*. The specific pattern used for generating the synthesized UDD is as follows:

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

The adoption of this standardized format offers a concise and comprehensive representation of the core elements and occurrences associated with a specific rule. To illustrate this, we provide an example using an IFTTT rule that consists of the following components:

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

The synthesized pattern for this rule is as follows:

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

This pattern offers a succinct and clear representation of the rule's components and their corresponding values, making it easy to understand the intended functionality. This holds true even after examining the original description:

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

4.3. Sample Generation

To configure an effective model, we undertook the creation of a new dataset, encompassing all possible types of UDDs that a user could generate for their automation rules. These UDD types fall into three distinct macro categories:

- ***Completely Consistent UDDs***: These UDDs are coherent, aligning perfectly with both the trigger and action components.
- ***Completely Inconsistent UDDs***: In contrast, these UDDs lack coherence with both the trigger and action components.
- ***Partially Consistent UDDs***: This category includes UDDs that exhibit coherence with either the trigger component or the action component, but not both.

To achieve this UDD diversification, we utilized the synthesizing strategy described in the previous section. Before delving into the strategy's adoption, we first defined the desired final structure of the dataset. Our objective was to construct a random sample of 20,000 entries, each comprising three key features: the UDD (description component), the synthesized pattern, and the corresponding label indicating the class of consistency between the UDD and its pattern.

After this, we embarked on generating different types of UDDs. For the first type of UDD (completely consistent), we randomly selected 5,000 rules with consistent descriptions from the initial dataset and synthesized their corresponding patterns from the rule's components. For the second type (completely inconsistent), we took 5,000 random rules and replaced each one's correct description with a different description that includes a distinct combination of trigger and action components. This shuffling mechanism introduces inconsistencies in the UDDs. It is important to note that the shuffling is applied only to the UDDs, while the patterns are synthesized with the correct components of the user's rule.

However, the most challenging aspect was defining samples for the last type of UDDs (partially consistent). To address this, we selected 10,000 random rules from the original dataset and divided them into two separate sets. This division enabled us to obtain 5,000 patterns with only the wrong trigger component (randomized from the other trigger components of the dataset) and another 5,000 patterns with only the wrong action component (randomized from the other action components of the dataset). These newly construed patterns were then used as input for the LLM Alpaca-Lora [2] [17, 32] to generate partially correct UDDs.

We adopted this approach due to the difficulty in designing precise instructions for a LLM. Instead, we opted for a single prompt, instructing the model to generate a textual description explaining the behavior of the automation rule

[2] https://crfm.stanford.edu/2023/03/13/alpaca.html

based on the input pattern containing the trigger and action components. The instruction prompt is the following:

> *Given an input sentence containing the trigger and action components of a trigger-action rule, execute the following instruction:*
> *"Generate a textual description explaining the behavior of the trigger-action rule."*

For example, given the following partially consistent pattern:

> ***IF** Current condition changes to (RSS Feed), **THEN** Post a tweet (Twitter)*

The model generates the corresponding UDD:

> *When the current condition changes to RSS Feed, a tweet is posted on Twitter.*

This prompt was used for both tasks, streamlining the process and ensuring consistent results. It is worth noting that, in the final dataset, we have the patterns synthesized with the correct components of the user's rule. The described process was solely for defining samples of partially correct user descriptions, and the generated UDDs were manually checked to ensure their correctness.

By incorporating these techniques and expanding the dataset to include a wide range of UDD types, we achieved a more robust and accurate set of data for the training phase.

4.4. Data Labeling

For the final stage of dataset composition, we proceeded to assign the appropriate labels to the UDD-pattern pairs. Building on our previous work [6], we had initially defined two semantic similarity classes:

- ***Contradiction***: This label denotes inconsistency between the UDD and the synthesized pattern, indicating that the UDD and the pattern do not align in their descriptions.
- ***Entailment***: This label signifies consistency between the UDD and the synthesized pattern, implying that the UDD's description is in agreement with the pattern.

However, this approach resulted in the exclusion of some possible scenarios, specifically the partial consistency UDDs, from the definition of a trigger-action rule description.

To address this limitation, we decided to define more appropriate labels for UDD-pattern pairs, taking into account the internal division of the dataset. As a result, we delineated the following four classes:

- ***"ee"***: This class denotes complete consistency between the UDD and the synthesized pattern, indicating that both the trigger and action components are accurately represented in the UDD.
- ***"cc"***: This class denotes complete inconsistency between the UDD and the synthesized pattern, indicating that neither the trigger nor the action components are correctly aligned in the UDD.
- ***"ec"***: This class denotes partial consistency between the UDD and the synthesized pattern, with a focus on the trigger component. Specifically, the trigger component in the UDD is correct, but the action component does not align with the pattern.
- ***"ce"***: This class denotes partial consistency between the UDD and the synthesized pattern, with a focus on the action component. Specifically, the action component in the UDD is correct, but the trigger component does not align with the pattern.

With the generation of our new dataset, we no longer require manual labeling since we have systematically produced the samples. Through the devised strategy and patterns, we are able to create diverse UDD-pattern pairs, covering various consistency scenarios, including completely consistent, completely inconsistent, and partially consistent cases.

The use of the LLM Alpaca-Lora enabled us to generate new UDDs that mimic user-generated descriptions with errors or partial consistencies. This procedure proved especially beneficial for creating UDDs falling into the "ec" and "ce" classes, where either the trigger or the action component was accurately represented, but the other exhibited inconsistencies. By automating the sample generation process, we have significantly reduced the manual effort involved in labeling the data. The resulting dataset contains a comprehensive representation of UDDs, covering a wide range of semantic similarities with their corresponding synthesized patterns.

With this augmented dataset, we can now proceed to train and evaluate our NLP classification models for the semantic consistency checking task more efficiently and effectively. The automated generation of samples not only saves time but also enhances the dataset's diversity, contributing to the overall robustness and accuracy of the trained models.

5. Classification Models

This section details the implementation of models for classifying the UDD-pattern pairs, the techniques used to develop each model, and the training phase setup.

We consider three different transformer-based models to assess the semantic consistency between UDDs and rule behaviors:

1. **BERT-based model**: This model is based on BERT [14]. The latter is an encoder-only model that utilizes deep bidirectional transformers and has been pretrained on a large corpus of data to create a powerful NLP language representation model.

2. **T5 model**: The T5 [29] model adopts a unique approach as an encoder-decoder model, learning to predict masked words in sentences through a corrupting span denoising objective.

3. **GPT-2 model**: The GPT-2 [28] model is an unsupervised generative language model developed by OpenAI. It operates as a decoder-only model based on the transformer architecture.

By employing these three transformer-based models, we aim to thoroughly analyze and compare their performance in determining the semantic alignment between UDDs and rule behaviors. The first model, based on the BERT architecture, involves an encoder-only approach and a feature extraction layer to represent input tokens. The second model, T5, utilizes both encoder and decoder components, offering insights into the performance of combined architectures. Finally, the third model, GPT-2, allows us to explore the capabilities of a purely generative approach.

In all models, features are treated as text. Moreover, before training the models, a pre-processing phase is performed to remove noise from UDDs. This includes operations such as normalization and lemmatization on the textual values.

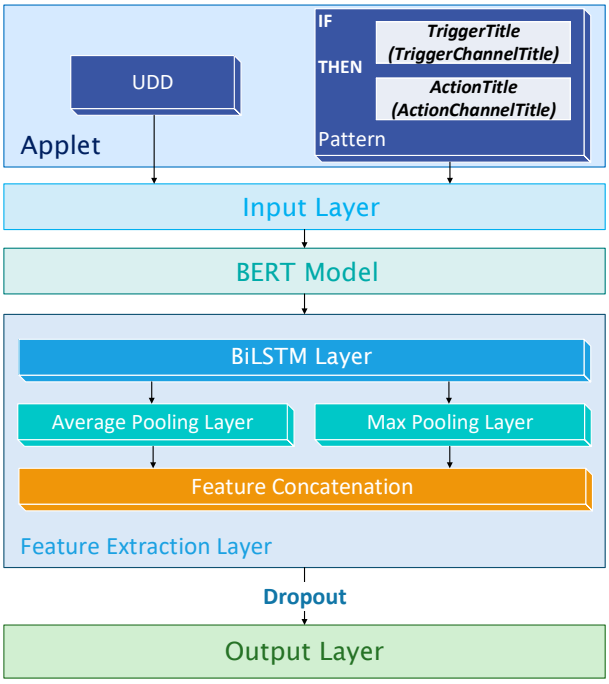

Figure 3: The architecture of the BERT-based model

5.1. BERT-based Model

The Bidirectional Encoder Representations from Transformers (BERT)-based model's architecture for classifying the semantic consistency of UDD-pattern pairs is illustrated in Figure 3. This model comprises interconnected components working together to achieve the objective. The initial component is the *Input Layer*, which encodes UDD-pattern pairs into numerical representations known as dense vectors. These dense vectors are then processed by the BERT language model, consisting of multiple Transformer Encoder Layers that generate contextual representations of each word in the input sequence using self-attention. Each layer produces dense vectors capturing different levels of syntactic and semantic information.

Next, the sequence obtained from the BERT model is passed to the *Feature Extraction Layer*, containing a *Bidirectional Long Short-Term Memory* (BiLSTM) Layer. The BiLSTM is designed to store past and future context, and its output consists of a sequence of vectors representing the hidden states at each time step. These hidden states are concatenated to create a representation capturing global features and dependencies.

The output from the BiLSTM Layer is processed through an *Average Pooling Layer* and a *Max Pooling Layer*, reducing dimensionality by aggregating information across the sequence. Average pooling calculates the average value of each feature, providing an overall representation and distribution. On the other hand, max pooling selects the maximum value from each dimension, highlighting salient features. Both pooling operations are concatenated through a *Feature Concatenation* module to create a comprehensive representation that captures overall context and important local details. A *Dropout* operation is applied before feeding the concatenated data to the *Output Layer*, randomly dropping out input features to mitigate overfitting.

The *Output Layer* utilizes the extracted features to evaluate semantic consistency between rule descriptions and corresponding patterns. It applies linear transformations to compute the final classification output of the model, determining whether the UDD-pattern pairs exhibit semantic consistency.

5.2. T5 Model

The architecture of the Text-To-Text Transfer Transformer (T5) model, designed specifically for the text classification task, is delineated in Figure 4. T5's approach to text classification is distinct from traditional methods that use separate encoder and decoder components. Instead, it transforms all tasks into a text-to-text format, where both the input and output are treated as text sequences. This allows T5 to handle different tasks by simply modifying the input and output representations.

The architecture begins with an *Input Layer* that takes the UDD-pattern pairs to be classified as input. The input text is processed and converted into a sequence of tokens, with each token representing a word or subword unit. These tokens are then embedded into a dense, continuous vector space using an *Embedding Layer*.

The embedded tokens enter the *Encoder Transformer Blocks*, which are responsible for processing the input text. Each encoder block consists of a multi-head self-attention layer, a feedforward neural network layer, and layer normalization. The self-attention layer allows the model to attend to different parts of the input text, capturing the relationships be-

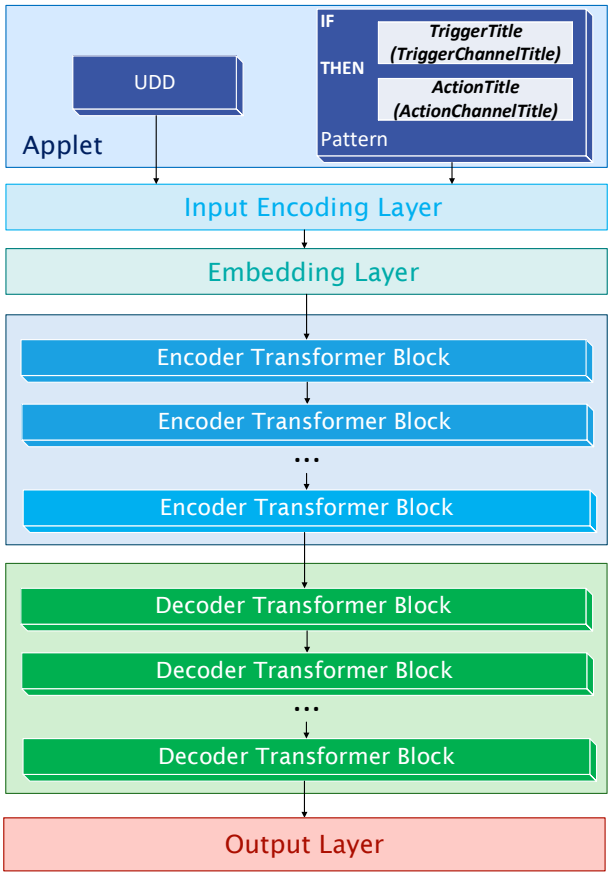

Figure 4: The architecture of the T5 model

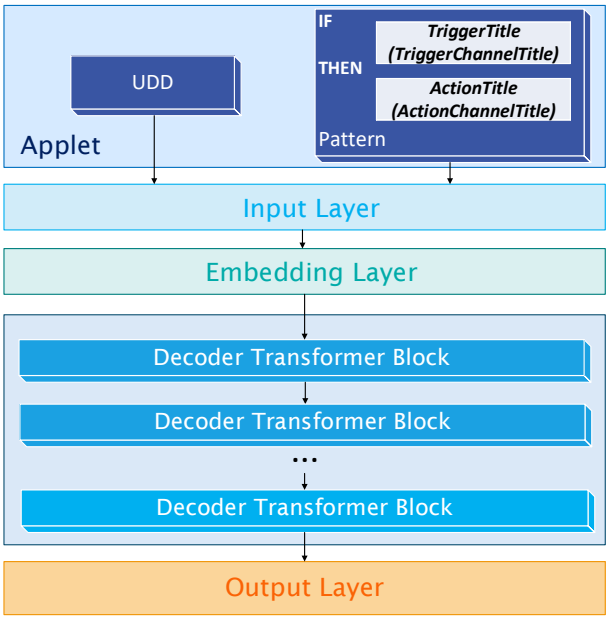

Figure 5: The architecture of the GPT-2 model

tween words and their context. The *Feedforward Neural Network* introduces non-linear transformations, and layer normalization helps stabilize the training process.

The output of the encoder blocks is a sequence of contextualized representations, with each token's representation containing information about its surrounding context. These representations are then passed to the *Decoder Transformer Blocks*. The latter further process the encoder's contextualized representations to generate task-specific outputs. Each decoder block has similar components to the encoder blocks, such as multi-head self-attention, Feedforward Neural Networks, and layer normalization. However, the decoder also includes cross-attention layers, allowing it to focus on both the input sequence and the task representation simultaneously.

Finally, the *Output Layer* receives the processed output from the last decoder block. This layer is customized to the specific text classification task and further processes the contextualized representations. It produces classification scores for each possible class, determining the predicted class for the input text based on the highest probability.

5.3. GPT-2 Model

The Generative Pre-trained Transformer 2 (GPT-2) model was originally designed for generating coherent and diverse text, but its capabilities have extended to include highly useful applications in text classification tasks as well.

The architecture of GPT-2, as depicted in Figure 5, is based on the transformer architecture, and it primarily leverages the decoder component of the transformer. Unlike encoder-decoder models such as T5, GPT-2 does not utilize the encoder part of the transformer. This design choice enables GPT-2 to excel in its primary function of generating text with contextual understanding.

In the context of text classification with UDD-pattern pairs as input, GPT-2's architecture begins with an *Input Layer*, which takes a sequence of tokens representing words or subword units. These tokens are then transformed into dense, fixed-dimensional vectors using an *Embedding Layer*. This mapping process places the tokens into a continuous vector space, facilitating the model's ability to capture semantic relationships and similarities between words.

The embedded tokens are then passed through a series of *Decoder Transformer Blocks*. Each transformer block is a stack of layers, consisting of a multi-head self-attention mechanism, which allows the model to attend to different parts of the input sequence and capture dependencies between words in the context of the entire sequence. Additionally, each block contains a *Feedforward Neural Network Layer*, which introduces non-linear transformations to the token representations, further enhancing the model's ability to model complex relationships.

In order to ensure stable training and facilitate faster convergence, each transformer block incorporates a normalization phase. This process normalizes the activations in each layer, enhancing the robustness of the optimization process.

As the input sequence progresses through the stack of transformer blocks, the model gains a deeper understanding of the context and relationships between the tokens. The fi-

nal transformer block produces a sequence of hidden states, where each hidden state corresponds to the encoded representation of its corresponding token.

For text classification tasks, the hidden states are then passed to the *Output Layer*, which performs a weighted combination of the hidden states and generates classification scores for each possible class. The class with the highest score is selected as the predicted class for the input text.

6. Models Evaluation

In this section, we present an analysis of the performances of the classification models. Specifically, we provide details on the experimental setup, the adopted metrics, and the results obtained from the experiments. The code of the software is publicly available on GitHub[3].

6.1. Evaluation Setup

In the evaluation phase, we trained the three NLP models using specific methodologies.

For the BERT-based model, we followed a two-step process. Initially, we froze all pre-trained layers and focused on training only the top layers. This allowed us to extract features by utilizing the representations of the pre-trained model. After feature extraction, we proceeded with an additional fine-tuning step. During this step, we unfroze the BERT model and retrained the entire architecture with a significantly low learning rate. The objective was to progressively adapt the pre-trained features to the new data, leading to enhanced model performance.

To pre-train and fine-tune the BERT-based model, we utilized Python libraries, specifically `Keras` and `TensorFlow`. We chose the "bert-base-uncased" variant, which has 12 transformer blocks, 768 hidden units, and 12 self-attention heads. It is designed to handle lowercase letters. The training set consisted of 13,999 samples, encompassing all four types of consistency classes. To optimize hyperparameters, we employed a validation set with 4,000 samples. The best hyperparameter configuration included 4 epochs, a batch size of 64, an epsilon set to 1e-5, and a maximum text length of 70. The model's performance was then evaluated using a test set of 2,001 UDD-pattern labeled pairs.

For GPT-2 and T5, we followed the same libraries and dataset sizes but made adjustments to hyperparameters. T5 was trained with 6 epochs, a batch size of 24, and a maximum text length of 70. On the other hand, GPT-2 was trained with 5 epochs, a batch size of 32, and a maximum text length of 60.

6.2. Evaluation Metrics

The performance evaluation of the proposed models involves several metrics, i.e., Accuracy, Precision, Recall, and F1-score. These metrics are computed based on the values of True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). The evaluation metrics can be expressed as follows:

[3] https://github.com/empathy-ws/TAP-Semantic-Consistency-Checking

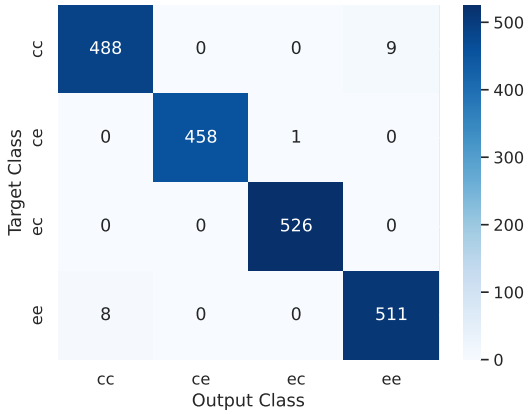

Figure 6: Confusion Matrix of the BERT-based model

- **Accuracy** is a measure of the overall correctness of a model's predictions, expressed as the ratio of the number of correctly classified samples to the total number of samples evaluated:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

- **Precision** is a measure of the proportion of true positive samples among all samples that the model identified as positive:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

- **Recall** is a measure of the proportion of true positive samples among all actual positive samples:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

- **F1-score** is the harmonic mean of Precision and Recall:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

6.3. Results and Discussion

In our investigation, we leveraged three transformer-based NLP models to evaluate the semantic consistency between UDDs and rule behaviors. The confusion matrices obtained from the classification results on the test set for each classification model are presented in Figure 6, Figure 7, and Figure 8, respectively. Additionally, Table 1 provides the resulting values of Accuracy, Precision, Recall, F1-score, and the average of the per-class metrics for each model.

The BERT-based model achieved the highest accuracy and precision values (99%), indicating its superior ability to make correct predictions overall. It excelled in identifying the ee and ec classes with high recall rates but encountered challenges in distinguishing class cc, leading to some misclassifications where ee was incorrectly predicted as cc. On

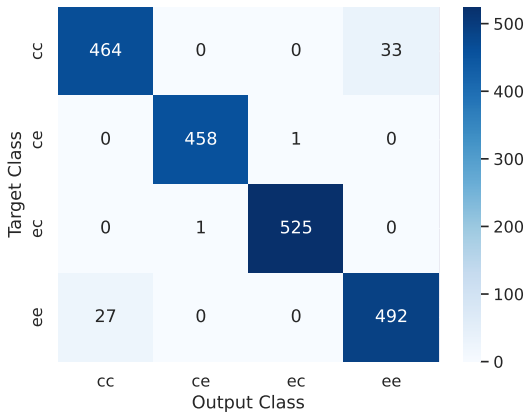

Figure 7: Confusion Matrix of the T5 model

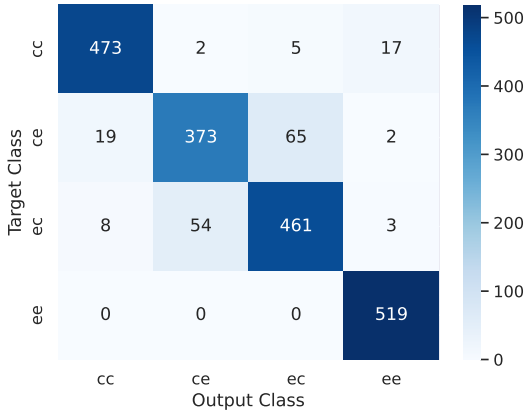

Figure 8: Confusion Matrix of the GPT-2 model

the other hand, the T5 model achieved an accuracy of 97%, exhibiting strong overall performance. Specifically, it performed well in identifying the ee and ec classes, evident from their recall values (Table 1). However, akin to the BERT-based model, the T5 model encountered difficulties concerning class cc, frequently leading to erroneous classification as an ee class. Finally, the GPT-2 model achieved an accuracy of 91% and exhibited the lowest discriminative ability among the three models. As highlighted by the confusion matrix (Figure 8), it faced notable challenges in differentiating class cc from other classes, increasing confusion between ec and ce. Despite these limitations, the GPT-2 model still produced mostly correct predictions.

The superior performance of the BERT-based model over the T5 and GPT-2 models can be attributed to the fundamental differences in their pre-training approach. In particular, BERT leverages the Masked Language Modeling (MLM) [14] technique during pre-training, where certain words in the input text are randomly masked, and the model is tasked with predicting these masked words based on contextual cues from the surrounding words. This process equips BERT

Table 1
Classification performances of the models on the test set

	Metric	cc	ce	ec	ee	Avg
BERT						
	Precision (%)	98	100	100	98	99
	Recall (%)	98	100	100	98	99
	F1-score (%)	98	100	100	98	99
	Accuracy (%)					99
T5						
	Precision (%)	95	100	100	94	97
	Recall (%)	93	100	100	95	97
	F1-score (%)	94	100	100	94	97
	Accuracy (%)					97
GPT-2						
	Precision (%)	95	87	87	96	91
	Recall (%)	95	81	88	100	91
	F1-score (%)	95	84	87	98	91
	Accuracy (%)					91

with a solid capability for acquiring contextual representations of words and comprehending intricate relationships between them within sentences. On the contrary, T5 adopts an innovative text-to-text approach [29] during its pre-training phase, where the input text serves as a description of a specific NLP task, while the output text represents the corresponding solution or result for that particular task. When focusing specifically on classification tasks, the MLM approach of BERT exhibits notable advantages. By predicting masked words, BERT gains a deeper understanding of how words are interconnected within a given context, resulting in the proficient classification of text. Instead, formulating classification tasks into the text-to-text format for T5 might not be as straightforward as employing BERT's masked language modeling. Indeed, ensuring that the task descriptions lead to accurate and effective classification can pose challenges and may necessitate meticulous formulation and experimentation to achieve optimal results. Furthermore, it is essential to consider the trade-off between specificity and generalization. While BERT's MLM approach enables it to focus on the contextual nuances of individual words, T5's text-to-text approach emphasizes generalization across diverse tasks. As a consequence, T5 might not capture certain task-specific nuances as effectively as BERT in certain classification scenarios. Finally, unlike the BERT and T5 models, the GPT-2 model adopts a unidirectional approach (i.e., it processes tokens in a left-to-right manner), which can result in a less comprehensive understanding of the input text. This unidirectional nature may have impacted its ability to effectively differentiate between the classes, particularly in cases where the context from the right side of the input text was crucial for accurate prediction.

The promising performance of models in checking semantic consistency between UDDs and rule behavior presents opportunities for further research and improvements in transformer-based NLP classification for similar tasks. Understanding the strengths and weaknesses of different transformer

models helps inform the selection of appropriate models based on the task requirements. As transformer-based NLP models continue to advance, they hold great potential in enhancing the accuracy and efficiency of various natural language understanding tasks, benefiting a wide range of applications in the domain of IoT device automation and beyond.

7. Conclusion and Future Work

This paper introduces an innovative approach to tackle the issue of potential inaccuracies and misleading information in UDDs shared on TAPs. We developed a new dataset that comprehensively covers different types of UDD scenarios, including consistent, inconsistent, and partially consistent descriptions. To achieve this, we leveraged an LLM to generate samples with partially correct UDDs, which reduced the manual workload and increased dataset heterogeneity.

The evaluation involved three NLP classification models: BERT-based, T5, and GPT-2. The BERT-based model underwent a two-step training process, where initially, the top layers of the pre-trained model were targeted for training, and then fine-tuning was conducted with the entire architecture. The T5 and GPT-2 models were also trained using the same dataset size as BERT. The experimental results on the IFTTT dataset, consisting of 20,000 labeled UDD-pattern pairs, demonstrated the effectiveness of the proposed models. The BERT-based model demonstrated remarkable performance, achieving an overall accuracy of 99%, complemented by precision, recall, and F1-score, all attaining a value of 99% as well. In contrast, the T5 model exhibited slightly inferior performance, with all evaluation metrics registering at 97%. The GPT-2 model yielded the least favorable results, scoring 91% for all evaluation metrics. This outcome accentuates the relatively weaker performance of the decoder-only architecture-based model in performing the classification task. Nevertheless, it is noteworthy that all models exhibited a high degree of reliability in discerning compliant UDDs from unrelated ones, establishing a robust and dependable method for conducting semantic consistency checks in TAPs.

In the future, we would like to consider the adoption of this approach in other TAPs beyond the IFTTT case study. This broader application would enable us to further validate the robustness and effectiveness of our classification models in different environments and contexts. Additionally, we aim to explore potential enhancements to the models, such as incorporating more advanced NLP techniques or leveraging larger and more diverse datasets for training. Furthermore, we believe that integrating user feedback and iterative improvements to the models would be valuable in optimizing the accuracy and reliability of the semantic consistency checking process. In addition, providing explainability to users has already been proven to increase users' trust in the systems in several domains [4, 8]. Thus, we think incorporating such a module would allow us to contribute to the evolution of TAPs, in terms of trust, security, and ease of use in defining automation rules while safeguarding users against potential risks arising from misleading or deceptive rule descriptions.

References

[1] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., Zhou, Y., 2017. Understanding the Mirai Botnet, in: Proceedings of the 26th USENIX Conference on Security Symposium, USENIX Association, USA. p. 1093–1110.

[2] Breve, B., Cimino, G., Desolda, G., Deufemia, V., Elefante, A., 2023a. On the user perception of security risks of tap rules: A user study, in: International Symposium on End User Development, Springer. pp. 162–179.

[3] Breve, B., Cimino, G., Deufemia, V., 2021. Towards a classification model for identifying risky IFTTT applets, in: Proceedings of the 2rd International Workshop on Empowering End-Users in Dealing with Internet of Things Ecosystems, pp. 33–37.

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

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

[6] Breve, B., Cimino, G., Deufemia, V., Elefante, A., 2023c. A BERT-based model for semantic consistency checking of automation rules. Proceedings of the 29th International DMS Conference on Visualization and Visual Languages .

[7] Breve, B., Cimino, G., Deufemia, V., Elefante, A., 2023d. On privacy disclosure from user-generated content of automation rules, in: Joint Proceedings of the Workshops, Work in Progress Demos and Doctoral Consortium at the IS-EUD 2023, pp. 1–5.

[8] Cascone, L., Pero, C., Proença, H., 2023. Visual and textual explainability for a biometric verification system based on piecewise facial attribute analysis. Image and Vision Computing 132, 104645.

[9] Celik, Z.B., Tan, G., McDaniel, P.D., 2019. IoTGuard: Dynamic enforcement of security and safety policy in commodity IoT, in: Network and Distributed System Security (NDSS) Symposium.

[10] Chiang, Y.H., Hsiao, H.C., Yu, C.M., Kim, T.H.J., 2020. On the privacy risks of compromised trigger-action platforms, in: Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25, Springer. pp. 251–271.

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

[12] Corno, F., De Russis, L., Monge Roffarello, A., 2020. HeyTAP: Bridging the gaps between users' needs and technology in IF-THEN rules via conversation, in: Proceedings of the International Conference on Advanced Visual Interfaces, pp. 1–9.

[13] Desolda, G., Ardito, C., Matera, M., 2017. Empowering end users to customize their smart environments: model, composition paradigms, and domain-specific tools. ACM Transactions on Computer-Human Interaction (TOCHI) 24, 1–52.

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

[15] Ding, W., Hu, H., 2018. On the safety of iot device physical interaction control, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pp. 832–846.

[16] Hsu, K.H., Chiang, Y.H., Hsiao, H.C., 2019. Safechain: Securing trigger-action programming from attack chains. IEEE Transactions on Information Forensics and Security 14, 2607–2622.

[17] Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang,

L., Chen, W., 2021. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 .

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

[19] Johnsson, B.A., Magnusson, B., 2020. Towards end-user development of graphical user interfaces for internet of things. Future Gener. Comput. Syst. 107, 670–680.

[20] Li, S., Xu, L.D., Zhao, S., 2015. The internet of things: a survey. Information systems frontiers 17, 243–259.

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

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

[23] Mi, X., Qian, F., Zhang, Y., Wang, X., 2017. An empirical characterization of IFTTT: ecosystem, usage, and performance, in: Proceedings of the 2017 Internet Measurement Conference, pp. 398–404.

[24] Naveed, H., Khan, A.U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., Mian, A., 2023. A comprehensive overview of large language models. arXiv preprint arXiv:2307.06435 .

[25] Neshenko, N., Bou-Harb, E., Crichigno, J., Kaddoum, G., Ghani, N., 2019. Demystifying iot security: An exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations. IEEE Communications Surveys & Tutorials 21, 2702–2733.

[26] Paternò, F., Santoro, C., 2019. End-user development for personalizing applications, things, and robots. International Journal of Human-Computer Studies 131, 120–130.

[27] Quirk, C., Mooney, R., Galley, M., 2015. 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), pp. 878–888.

[28] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al., 2019. Language models are unsupervised multitask learners. OpenAI blog 1, 9.

[29] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research 21, 5485–5551.

[30] Surbatovich, M., Aljuraidan, J., Bauer, L., Das, A., Jia, L., 2017. Some recipes can do more than spoil your appetite: Analyzing the security and privacy risks of IFTTT recipes, in: Proceedings of the 26th International Conference on World Wide Web, p. 1501–1510.

[31] Tian, Y., Zhang, N., Lin, Y.H., Wang, X., Ur, B., Guo, X., Tague, P., 2017. Smartauth: User-centered authorization for the internet of things, in: Proceedings of the 26th USENIX Conference on Security Symposium, USENIX Association, USA. p. 361–378.

[32] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al., 2023. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 .

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

[34] Wang, X., Li, J., 2013. Detecting communities by the core-vertex and intimate degree in complex networks. Physica A. 392, 2555–2563.

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

[36] Yao, Z., Li, X., Gao, J., Sadler, B., Sun, H., 2019. 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 Press.

[37] Yusuf, I.N.B., Jiang, L., Lo, D., 2022. 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, Association for Computing Machinery, New York, NY, USA. p. 99–110.

[38] Zeng, E., Mare, S., Roesner, F., 2017. End user security and privacy concerns with smart homes, in: Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017), USENIX Association, Santa Clara, CA. pp. 65–80. URL: https://www.usenix.org/conference/soups2017/technical-sessions/presentation/zeng.