# Hybrid Prompt Learning for Generating Justifications of Security Risks in Automation Rules

BERNARDO BREVE, GAETANO CIMINO, and VINCENZO DEUFEMIA, University of Salerno, Fisciano (SA), Italy

Trigger-action platforms (TAPs) enable users without programming experience to personalize the behavior of Internet of Things applications and services through IF-THEN rules. Unfortunately, the arbitrary connection of smart devices and online services, even with simple rules, such as "IF the entrance *Netatmo Wheather Station* detects a temperature above $30°C$ ($86°F$) THEN open the shutters in the living room," might expose users to potential security and privacy risks (e.g., the execution of the previous rule might provide an easy entry point for thieves, especially during the summer vacation period). The goal of our research is to make the users capable of understanding and mitigating the threats and risks associated with the execution of IF-THEN rules. To this end, we define a new challenging task, namely generating post hoc justifications of privacy and security risks associated with automation rules, and propose a novel natural language generation strategy based on hybrid prompt learning producing justifications in the form of real-life threat scenarios. The proposed strategy allows for prompt customization with task-specific information, providing contextual details enabling to grasp the nuances and subtleties of the domain language, resulting in more coherent justifications. The experiments conducted on the if-this-then-that (IFTTT) platform show that our method produces effective justifications, improving the explainability of discrete and hybrid prompting methods up to 27% in BLEURT score. The code of the software is publicly available on GitHub[1].

CCS Concepts: • **Computing methodologies** → **Natural language generation**; • **Security and privacy** → *Usability in security and privacy*;

Additional Key Words and Phrases: Explainable privacy and security, natural language generation, pre-trained language model, prompt learning, transformer, trigger-action rules

**ACM Reference format:**
Bernardo Breve, Gaetano Cimino, and Vincenzo Deufemia. 2024. Hybrid Prompt Learning for Generating Justifications of Security Risks in Automation Rules. *ACM Trans. Intell. Syst. Technol.* 15, 5, Article 103 (November 2024), 26 pages.
https://doi.org/10.1145/3675401

[1]https://github.com/empathy-ws/Justify-Harms-of-Trigger-Action-Rules

This work is supported by Italian Ministry of University and Research (MUR) under PRIN 2017 "EMPATHY: Empowering People in deAling with internet of THings ecosYstems" Grant No.:2017MX9T7H and by European Union-NextGenerationEU under PRIN 2022 PNRR "DAMOCLES: DetectionAnd Mitigation Of Cyber attacks that exploit human vuLnerabilitiES" Grant No.:P2022FXP5B.

Authors' Contact Information: Bernardo Breve, University of Salerno, Fisciano (SA), Italy; e-mail: bbreve@unisa.it; Gaetano Cimino, University of Salerno, Fisciano (SA), Italy; e-mail: gcimino@unisa.it; Vincenzo Deufemia (corresponding author), University of Salerno, Fisciano (SA), Italy; e-mail: deufemia@unisa.it.


ACM 2157-6912/2024/11-ART103
https://doi.org/10.1145/3675401

## 1 Introduction

The revolution of the **Internet of Things (IoT)** has turned everyday objects into their smart, interconnected versions [2]. This technological shift poses significant challenges in enabling users to interact with such devices [15]. The trigger-action programming paradigm was introduced as a solution to address these challenges, empowering users of all technical proficiency levels, including novices, to create event-driven behaviors to automate the interaction between devices within the smart environments [19, 34]. Such behaviors are typically structured as IF-THEN rules, where the trigger condition activates a specific action, and are enabled by **Trigger-Action Platforms (TAPs)**, which are designed with user-friendly interfaces, allowing users to create, activate, and share their rules. By using TAPs, users can avoid the manual labor of controlling devices and appliances by themselves. For instance, a rule, such as "*IF I leave my house, THEN turn off the light in the living room*" simplifies the task of manually turning off the lights every time they leave the house. The process of obtaining such automation behaviors has been further simplified by solutions that suggest and, in some cases, automatically define rules based on user preferences [13, 58, 62].

Unfortunately, the creation of automation may lead to unsafe behaviors, risking user security and privacy. For instance, the aforementioned rule could pose a significant security threat, as predictably turning off lights may signal an empty house to potential thieves. Despite available solutions identifying such risks [7, 17, 52], an open issue persists as the current state-of-the-art fails to provide comprehensive feedback on identified risks [6, 56]. To prevent unintended harm from unsafe individual rules, users need intuitive explanations of sources of instability, enabling them to take appropriate countermeasures and increase confidence in TAPs. On the other hand, existing approaches clarify the risks associated with the conflicting execution of rule sets through graphical models [51, 56], demanding technical proficiency that non-expert users lack, and requiring a thorough understanding of rule-specific intricacies due to their nuanced nature.

Viganò and Magazzeni [50] investigated various methods for explaining the reasoning behind security and privacy issues in generic systems. They proposed generating explanations at varying levels of granularity depending on the stakeholders' needs. For instance, technical experts might benefit from detailed explanations like explanation trees, attack trees, and formal languages. On the other hand, non-expert users would require explanations with a higher level of abstraction to facilitate comprehension, such as natural language explanations that are easily understood by all user types. As a consequence, we focus on the problem of generating textual explanations for the risks associated with trigger-action rules by using **Natural Language Processing (NLP)** techniques.

Among the different approaches for generating textual explanations [32, 59], **Language Models (LM)** represent a suitable solution to generate structured text that establishes a relationship between information on automation rules and the harm caused. In particular, *prompt learning* is gaining popularity as a training paradigm for pre-trained LM, especially in scenarios with limited or no labeled data [25]. The key idea behind prompt learning is to guide the pre-trained LM toward solving an end task by encoding training and test samples with a prompt. In this manner, we can take advantage of the pre-existing knowledge in the model and direct its behavior to generate the desired output. Three main approaches exist to generate suitable prompts: (1) *discrete prompt learning* involves creating *hard prompts*, which are textual sentences that use words from pre-trained LM datasets and do not add new parameters to the model [23], (2) *continuous prompt learning* uses *soft prompts*, which are embedded in a continuous space and fine-tuned using training samples [21], and (3) *hybrid prompt learning* combines both hard and soft prompts to enable pre-trained LM to perform tasks by inserting tunable embeddings into hard prompts [20, 25, 27].

To address the above issue, we propose a novel method to generate post hoc natural language justifications for security risks associated with automation rules. For sake of clarity, although the terms "justification" and "explanation" are often used interchangeably, we use the term "justification" as defined in [5]. Specifically, a justification is an account of why a decision is appropriate without explaining the specific mechanism behind it. The proposed approach exploits a **hybrid prompt learning** strategy to justify why an automation rule might cause specific harm to the user in terms of real-life threat scenarios. By presenting concrete examples of how the rules can be misused or exploited, users gain a better understanding of the importance of taking precautionary measures and being aware of potential risks. For instance, a rule that automatically uploads every photo taken on a smartphone to a social network might pose a risk to the user's privacy. In this case, a suitable justification would be: "*This rule could be harmful because if the user takes an embarrassing photo with the camera, it will be automatically uploaded to the social network, creating unpleasant situations.*"

Our proposal also introduces a novel strategy to generate soft prompts in the context of trigger-action rules. In fact, in such a domain the choice of trigger and action is subject to the selection of related components, called *channels*, denoting the company in charge of managing the smart device or the web service, e.g., Philips Hue for smart bulbs or Facebook regarding the interaction of posts uploaded to the social network. The generation of embeddings for channels is a particularly complex task because of the unlikelihood that their functionalities can be properly described within the vocabularies of pre-trained LM. Thus, the strategy we devised in this work, called *Channel2Vec*, enables the generation of soft prompts consistent with the domain of TAPs and trigger-action rules, making it possible to relate the embeddings of channels according to their functionalities thanks to the combination of a crawling system and a customized version of the Skip-Gram neural network model. These soft prompts are then combined with manually-defined hard prompts, creating hybrid prompts feeding pre-trained LM.

The proposed hybrid strategy is evaluated on **If-This-Then-That (IFTTT)**[2], a popular platform for self-automating IoT services, known for its compatibility with a wide range of services and devices. In our analysis, we collect justifications for 3,709 IFTTT rules, also known as "applets," to train LM relying on the Transformer architecture [49]. Specifically, we represent applet information as discrete tokens and employ the Channel2Vec embedding strategy to map channel names to continuous tokens. Experimental results demonstrate a significant improvement in the explainability of LM using our hybrid strategy compared to six baseline models relying on discrete and hybrid prompt learning, resulting in **Bilingual Evaluation Understudy with Representations from Transformers (BLEURT)** scores increasing by up to 27%. Moreover, readability analysis of the generated justifications suggests that users with a secondary school education can easily comprehend them. To further gauge the effectiveness of the proposed strategy, we conducted a comparative evaluation with two commercial **Large Language Models (LLMs)**, namely ChatGPT-3.5 and Claude.

The contributions of this article are summarized as follows:

- —We propose a hybrid prompt learning strategy to generate post hoc natural language justifications describing real-life threat scenarios of trigger-action rules.
- —We address the limitation of using discrete tokens with rule information that lack semantic meaning, by introducing Channel2Vec, a new approach for creating continuous tokens capturing domain-specific relationships between words.
- —We collected a justification dataset in the TAP domain, publicly available to other researchers.

[2] https://ifttt.com

—We evaluated the proposed approach on IFTTT demonstrating its superiority over discrete and hybrid prompt learning baselines. The experimental results show that the LM based on Phi [24] achieves the best performance in explainability and readability.

The remainder of this article is organized as follows. In Section 2, we review studies employing prompt learning and delve into methodologies associated with explainable security within the TAP domain, as well as the generation of natural language justifications. In Section 3, we formulate the considered problem and present the details of our methodology. In Section 4, we describe the experiments conducted to evaluate the benefits of the proposed approach and show the results in Section 5. Finally, Section 6 concludes this article and points out directions for future work.

## 2 Related Work

In the following, we survey relevant studies related to our work, starting with a thorough exploration of past research using prompt learning approaches. We then analyze efforts focused on clarifying security and privacy concerns in the TAP domain, alongside methods for generating natural language justifications.

### 2.1 Prompt Learning

LM are often pre-trained with objectives that differ from those of downstream tasks. Prompt learning mitigates this difference by formulating downstream tasks as LM problems incorporating language prompt templates [25]. The crux of prompt learning lies in the design of prompt templates, which can be broadly categorized into discrete, continuous, and hybrid types [25]. Discrete templates consist of existing natural words and heavily rely on human experiences [8, 44, 46]. For instance, Petroni et al. [38] manually crafted templates to prompt pre-trained LM in generating relational knowledge. Continuous and hybrid templates go beyond relying solely on human-interpretable natural language by introducing soft prompts with continuous vectors, which can be optimized by pre-trained LM [23, 27, 41]. For instance, Liu et al. introduced a continuous prompt tuning model named P-tuning, optimizing fill-in-the-blank prompts in a continuous space [27]. Similarly, Li and Liang proposed a method for prompt tuning using a textual prefix, wherein they prepend task-specific "soft tokens" (prefix) to the source text and tune the hidden states of only these tokens [23]. Qin and Eisner advocated for prompt learning in relation extraction tasks using data-dependent mixtures of prompt templates and parameters [41].

In this work, we introduce a novel context-aware approach that optimizes prompt encodings based on trigger-action rule contexts. Notably, we devise a hybrid prompt construction strategy to facilitate the natural language justification generation task in the TAP domain. Our approach involves serializing channel names with soft prompts and combining them with different hard prompts to obtain task-specific inputs.

### 2.2 Explainable Security in the TAP Domain

The term explainable security has been introduced in [50] as an extension of the explainable Artificial Intelligence program focusing on bridging the gap between the *actual* and the *perceived* security. In fact, XSec research field focuses on providing explanations that increase user confidence in using the system, rather than explanations that aim to build trust by revealing the inner workings of the system [40] (i.e., *explanation-for-confidence* rather than *explanation-for-trust*).

Several studies in the TAP domain focus on explaining security concerns. IoTSan [33] leverages model checking to detect inconsistencies within IoT systems with respect to a set of safety properties. In addition to the model checking phase, the authors have included a module called "Output Analyzer," which examines and authenticates logs, categorizing any violations as either

misconfigurations or behaviors caused by malicious applications. The output of this analysis is then presented to users as recommendations to either uninstall the app or change its configuration. A3ID [56] identifies implicit interferences among automation rules, which occur when multiple rules are triggered simultaneously and result in conflicting outcomes for the system. To assist users in comprehending the nature and underlying reasons for the interference issue, the authors provide explanations designed by means of a tree of concepts consisting of: (1) the scope of the actuators, (2) the impact of the actuators on the environment, (3) the functionalities they offer, and (4) the type of interference produced. Similarly, ProvThings [54] analyzes IoT apps and device APIs to identify faulty relationships between them. The authors aim at providing holistic behavioral explanations via a provenance graph illustrating a complete overview of the interactions enforced between rule components. FexIoT [51] detects vulnerable interactions among rules by modeling rule interplay through **Graph Neural Networks (GNNs)**. To protect user privacy, the authors develop a federated GNN framework with layer-wise clustering that enables prediction of vulnerable interactions without requiring users to disclose local information. Upon detecting a vulnerable graph, FexIoT employs a cause analysis method using Monte Carlo beam search and SHapley Additive exPlanations to uncover the likely causes of the vulnerable interactions by considering IoT dependency relationships. The goal is to identify the most probable subgraph responsible for the model's prediction, where each node represents a rule behavior and each edge represents the interaction between two rules. This provides users insight into potential risks by examining the paths within the implicated subgraph. Similarly, Glint [52] uses GNNs for real-time detection of interactive threats in closed-source smart home platforms, conceptualizing the task as a supervised graph representation learning problem. During the offline model training stage, Glint builds a GNN-based threat detector with an interaction graph dataset. During online use, it dynamically creates an interaction graph from event logs for real-time analysis. Existing GNN explanation tools, like SubgraphX [60], can be used to help identify potential threat causes. HAWatcher [17] employs a semantic-aware strategy for detecting anomalies and attacks. It leverages semantic information, including automation rules and device relations, to extract explainable correlations describing device and event relationships. Anomalies are discrepancies found when comparing simulated states with the actual device states. FaaS [26] is a non-intrusive forensic solution for smart homes that explains security risks. It collects device events from network traffic to build provenance graphs showing causal relationships. Policies specify expected behaviors in smart homes from a security perspective. With established policies and home models, an analysis is conducted to assess models against policies, thereby discerning anomalous situations. The explanation of security risks is accomplished by tracing identified risks back to their root causes, providing elucidation on the nature of any ensuing attacks, intrusions, or errors.

Unlike prior work explaining rule interferences [33, 51, 52, 54, 56], and forensic policy violations in smart homes [26], we focus on elucidating inherent risks of individual rules. Moreover, while previous solutions aim to improve user confidence via interpretable knowledge, their applicability to non-expert TAP users is questionable [53, 66]. For instance, graphical model explanations from FexIoT may be less effective for non-experts, especially when vulnerable interactions depend on intricate factors like environmental effects [56]. This is because users need to engage in reasoning and make assumptions about potential interference scenarios. Therefore, we propose generating natural language justifications depicting real-life threat scenarios, leveraging empathy and aligning with users' mental models to enhance comprehension of rule risks.

Unlike HAWatcher, which analyzes semantics after the installation of a rule, our solution provides users with an explanation of potential privacy and security risks in automation rules at the deployment stage. Moreover, while HAWatcher offers explainable correlations, it lacks insight into

the causative factors behind anomalies. In contrast, our natural language justifications provide users with a precise understanding of threat origins.

### 2.3 Generation of Natural Language Justifications

The generation of justifications using natural language has gained increasing interest in recent years, particularly in the generation of personalized recommendations [22, 32, 59, 64]. Using natural language sentences for justification is ideal for this task due to the abundance of textual data in the form of reviews and advancements in natural language generation techniques [22]. Employing natural language justifications proved itself a valuable means for enforcing trustworthiness and transparency in crucial domains, such as e-health, and supporting decision-making processes [63]. State-of-the-art solutions fulfilled the issue by employing different methodologies, such as the involvement of recurrent neural networks [14, 36, 64], a combination of NLP-based techniques [32], and the definition of pipelines comprising post hoc algorithms [59]. In particular, Zhao et al. [64] introduced an approach for providing human-like justifications to a song recommendations system, with the goal of improving users' awareness. The authors aimed to train a LM to produce reasons for recommending a specific song. By incorporating factors, such as personal emotion, along with information about the recommended song, the proposed solution is able to generate a personalized justification for choosing it. Park et al. [36] expands the domain of query answering on images with the definition of a model capable of jointly providing visual and textual justifications, which serve to motivate the query output.

Yera et al. [59] investigate the use of various post hoc methods for explaining recipe recommendations. These methods were adapted from literature to the recipe domain, and include a simple-to-explain recommendation algorithm, global and local explanation mining algorithms based on discovering association rules, and a model-agnostic approach through learned surrogate models. The primary goal of the authors is to provide natural language justifications for why the recommended recipes are enjoyable, while also considering the influence of nutrition-aware criteria on the explanatory capacity of the methods.

The framework presented by Musto et al. [32] is capable of generating natural language justifications for recommendation algorithms on a wide range of topics. The authors offer three distinct solutions to tackle this task: the first approach utilizes sentiment analysis and NLP techniques to identify significant features in reviews. The second approach builds upon the first by incorporating more advanced techniques like automatic feature extraction and text summarization. Finally, the third approach produces context-aware justifications by developing a unique vocabulary for each context to provide justifications that vary based on the specific contextual circumstances.

Recently, Li et al. [22] proposed the generation of justifications using LM trained on prompts. In particular, they experimented with both discrete and continuous prompt learning approaches, and found that the latter performed better. They optimized the soft prompt through a two-step training strategy, initially fine-tuning prompt parameters alone and later refining all parameters together. Xie et al. [57] focus on the generation of informative and factually grounded justifications for recommender systems. Their modular framework includes a personalized retriever, rating prediction module, and justification generation module. A **Bidirectional Encoder Representations from Transformers (BERT)**-based personalized retriever creates a latent query vector from historical reviews, integrated into a matrix factorization model for rating predictions. Justification generation uses a keyword-guided question answering strategy, retrieving relevant reviews through the latent query and using a fine-tuned GPT-2 model to generate informative keywords. These, along with the retrieved content and a specific question, arranged within a template consisting of hard prompts, are input into a question-answering model for the final rationale.

Our proposal is to use LM to generate personalized texts based on rule information and associated harms, avoiding post hoc justifications from fixed data sources [32, 59]. Such solutions risk oversimplifying scenarios, leading to potential inaccuracies based on pre-existing beliefs. LM, in contrast, can generate unique and never-seen-before texts, fostering novel ideas and creative scenario approaches. Moreover, while prior works relied on recurrent attention networks [36, 64], recent advances in pre-trained Transformer-based LM offer several advantages, including their ability to capture complex and diverse patterns in language and reduce labeled data required for fine-tuning. Finally, with respect to the approaches that exploit continuous or discrete prompt learning to generate justifications for recommended items [22, 57], we propose a hybrid prompt learning strategy for optimal use of input components in automation rules.

## 3 Methodology

In this section, we introduce the problem and describe the proposed methodology for generating post hoc natural language justifications that explain with a real-life threat scenario the reason why an automation rule might cause specific harm.

### 3.1 Problem Formulation and System Overview

The capability to create trigger-action rules within the realm of TAPs is reliant on the availability of an interface service provided by companies for IoT devices and web-based platforms, referred to as *channels*. Before establishing an automation rule, the selection of two channels is required: one for the event that triggers the rule and another for the corresponding action to be carried out. More formally, given a set of channels $C$, where each channel $c \in C$ provides a set of functionalities $TF(c)$ that can act as triggers and a set of functionalities $AF(c)$ that can act as actions, an automation rule $R$ can be represented as a quadruple $R = (c_i, c_j, t_{c_i}, a_{c_j})$, where $c_i, c_j \in C$ represent the Trigger and Action channels, respectively, identified as a unique word specifying their names, whereas $t_{c_i} \in TF(c_i)$ and $a_{c_j} \in AF(c_j)$ represent the Trigger and Action functionalities, respectively, specified as sequences of words, and selected from the pools made available by $c_i$ and $c_j$. As an example, the automation rule whose behavior entails the synchronization of any attachments from an incoming e-mail to a Google Drive folder can be represented by the quadruple ("*Gmail*," "*Google Drive*," "*Any new e-mail in your inbox with an attachment*," "*Upload file to Google Drive*").

In the considered natural language generation task, we assume a user has created an automation rule $R$ using a TAP, and an algorithm has identified a potential harm $H$ related to the execution of $R$. The harm could be represented as a label, such as the output of a classification model, or as a textual description, such as "loss of sensitive data" or "routine exposure." The goal is to generate a sequence of words $J = (j_1, j_2, \ldots, j_N)$ describing a scenario of execution for $R$ that can cause harm $H$. Text generation LM maximize $P(J \mid R, H)$, i.e., the probability of producing $J$ given the information on $R$ and $H$.

The proposed methodology requires a dataset $D$ consisting of $(R_i, H_i, J_i)$ triplets employed to guide LM in constructing comprehensible justifications. Clearly, creating a dataset of justifications requires a significant amount of human effort, making it difficult to acquire a large dataset for training LM. However, recent advancements in NLP have enabled the achievement of effective results with smaller amounts of data. For example, pre-trained LM trained on vast amounts of data can be fine-tuned for specific tasks to achieve state-of-the-art performance [55]. Therefore, we can leverage pre-trained LM to generate high-quality justifications in the TAP domain.

The process of generating a post hoc justification for a harmful rule is illustrated in Figure 1. The *Rule Definition* and *Harm Identification* modules involve defining an automation rule with the help of a TAP and identifying any privacy and security concerns. For example, in the figure, the rule $R$ is triggered by a new e-mail with an attachment from the Gmail channel, and the action is to

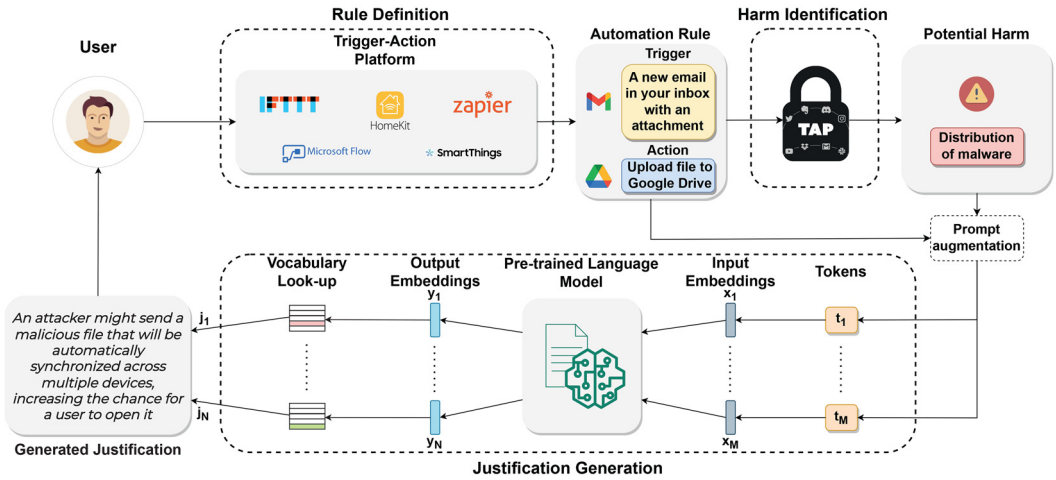


Fig. 1. The proposed process for generating justifications concerning harmful rules.

upload the attachment to the Google Drive channel. The harm identification algorithm identifies potential harm $H$ associated with this rule, such as "*Distribution of malware.*" A prompting function is applied to $R$ and $H$, which combines them with prompt tokens in a template. The *Justification Generation* module then uses this template to generate a justification, such as "*An attacker might send a malicious file that will be automatically synchronized across multiple devices, increasing the chance for a user to open it.*"

It is worth noting that the proposed methodology is agnostic with respect to the algorithm used to identify harms. Indeed, it could be an Artificial Intelligence-based classifier [7], a system that performs information-flow analysis [35] or model checking [9].

### 3.2 Generating Justifications based on Hybrid Prompt Learning

In the following, we introduce the proposed hybrid prompt learning strategy and the fine-tuning approaches for pre-trained LM.

*3.2.1 Hybrid Prompt Learning for Automation Rules.* An initial method to generate justifications for harmful rules is the exploitation of discrete prompt learning. Typically, when using a pre-trained LM, the embedding layer (denoted as $E$) is used to convert the input token sequence, represented as $X = (x_1, \ldots, x_N)$, into a set of embeddings $\{E(x_1), \ldots, E(x_N)\}$. In a downstream task, $X$ is conditioned by a specific context $CO$, which influences the model's behavior in producing the target sentence $J$. Instead, discrete prompt learning uses a prompting function that arranges $CO$, $J$, and a prompt $P$ into a template $T$. In our scenario, the prompt $P$ is made up of additional natural language tokens, while the context $CO$ consists of the rule information $R$ and the associated harm $H$. These are the key elements that guide the model in generating the justification $J$, which represents the target sentence.

*Hard Prompt.* To make prompts clear, we define them in a format similar to how a human would explain a problem. In particular, we consider two different *prefix* prompts [25], where the input is placed entirely before the slot $[J]$ to be filled. The first is named *span-infilling prompt* and is a straightforward sequence of natural language tokens following the encoded rule information:

[R] . This rule might cause a [H] harm because [J]

On the other hand, it has been proven that using the question-answering format can be very useful in transferring linguistic or other knowledge learned from one task to a related task [18]. For this reason, we also propose a *question-answering prompt* as it is possible to convert a non-question-answering task into an equivalent question-answering form [28]. Again, we deal with a prefix prompt since it is necessary to continue a string prefix, but the encoded rule information is placed in the middle of the prompt. Here, the prompting function receives as input the encoded rule $R$ and harm $H$, and outputs the sentence:

Why might rule [R] cause a [H] harm? [J]

In both cases, prompts are fixed, and training and test samples share the same prompt. The hypothesis is that the additional natural language tokens force the model to compute similarities between the input text and the words in the human-written templates, suggesting the subset of the vocabulary that provides better information in generating justifications.

Upon closer examination of rule characteristics, we found that, despite their simplicity, hard prompts have weaknesses. Specifically, representing channel names with discrete tokens may result in the loss of semantic information. In fact, channels are fundamental components in the TAP domain, helping determine a rule's execution context. For instance, if a rule uses "Philips Hue" as a channel for a trigger or action, it is clear that the rule is affecting a physical device. On the other hand, if the rule uses "Facebook," we can deduce that software-based operations will occur.

Pre-trained models often use *subword tokenization* to effectively deal with both common and infrequent words [30]. However, this method may result in the breaking down of channel names into meaningless segments, thereby exacerbating the challenge of comprehending the context in which the rules are executed. For example, the channel "500px" could be tokenized into ["500," "p," "x"], making it even more challenging to understand that it refers to a global online photo-sharing platform. A straightforward method would be to consider channel names as distinct words and incorporate them into the model's vocabulary. However, as the number of channels provided by TAPs grows rapidly, this method can cause a significant increase in the vocabulary size, resulting in a substantial computational load on the LM during the softmax operation used to generate a word at each step. Furthermore, pre-trained models utilize contextualized embeddings to represent words [37, 49], which capture contextual information in the text. This method is more efficient than using static word embeddings because it can generate different representations for words with multiple meanings. For instance, the word "mouse" can refer to a small rodent or a computer peripheral, and a pre-trained model can distinguish between these meanings based on the context. However, identifying the functionalities of a channel using this approach can pose a challenge in the TAP domain, as certain words may obscure the context. In fact, since channels might be words not known to the pre-trained LM, it is not certain that their representation is inferred only from words consistent with the exposed functionalities. For instance, consider the rule "*IF I publish a new post with a specific hashtag, THEN turn off the lights in the bedroom*" that enables a user to turn off the lights with a goodnight post. This rule can be defined using *Facebook* and *Philips Hue* as trigger and action channels, respectively. In this case, since the word "post," recalling a social media post, falls within the context of the Philips Hue channel, it may be considered in the definition of the channel's embedding, potentially undermining the rule's execution context. Ideally, the representation of Philips Hue should only include words, such as "lights" and "turn off" as it enables effective management of smart bulbs.

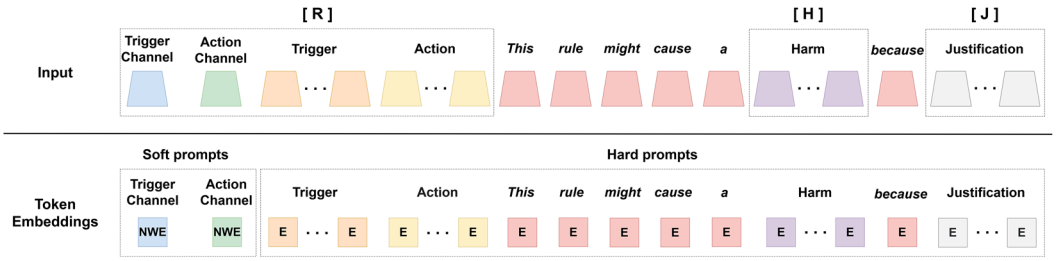


Fig. 2. Embeddings for the span-infilling prompt used in the hybrid prompt learning-based strategy.

*Hybrid Prompt.* The goal is to create embeddings for channels that are similar if they offer the same functionalities, such as camera or light services. In this way, during the generation phase, the model can more accurately deduce the context in which a rule is executed by considering channels similar to those involved in the rule. This is achieved by using soft prompts, where channels are represented as continuous tokens rather than discrete tokens. As an example, in Figure 2 we represent rule trigger, action, harm, justification, and prompt tokens as discrete tokens, obtaining their embeddings through the pre-trained LM embedding layer. Additionally, we consider rule channels as continuous tokens and derive their embeddings through neural word-embedding models. This results in a *hybrid* prompt that enables us to grasp both the rule's context of execution and the rule behavior.

Training soft prompts requires addressing two challenges [27]: (1) they cannot be randomly initialized and then optimized using stochastic gradient descent as it may lead to sub-optimal results [1], and (2) their values should be dependent on each other. Our approach addresses the interdependence among channels by considering the functionalities they expose, and avoids the issue of random initialization by generating their embeddings using a strategy inspired by Skip-Gram of *Word2Vec* [31]. The latter is a family of shallow, two-layer neural networks that can learn word embeddings of a vocabulary, ensuring that similar words have similar representations. The key concept behind this is that a word can be represented based on its context, i.e., the words surrounding it in documents. Therefore, words with similar contexts should have similar meanings.

The Skip-Gram model starts by assigning randomly generated embeddings to each word in the vocabulary. The goal is to improve these embeddings by predicting the words that appear in the context of a designated target word. To train the model, a dataset of pairs is used, where each pair consists of a *target word* and a *context word*. These pairs are created by analyzing existing documents, and the context window size determines how many words are considered before and after the target word. Specifically, the Skip-Gram model is fed with pairs (*target word*, *context word*), along with a label indicating whether the context word is contextually relevant (**Positive Input Sample (PS)**) or irrelevant (**Negative Input Sample (NS)**) to the target word. During the training process, the model works to improve its accuracy through multiple iterations and backpropagation steps. As a result, the embeddings become increasingly precise. Ultimately, the model's predictions are discarded, and only the optimized embeddings are used.

In the considered problem, vocabulary $V$ contains channel names of a TAP, aiming to model their embeddings based on their functionalities. However, generating a labeled dataset, as with the Skip-gram model, is not practical in this scenario. Channels appearing in the same context do not necessarily share functionalities, leading to unreliable samples. An example is depicted in Figure 3, where using a context window size of four and the source text "*Yesterday I posted a photo of my new Philips Hue bulb kit on Facebook*," we get the words "photo," "of," "my," "new," "bulb," "kit," "on" and "Facebook" as the context of "Philips Hue." The pairs (*Philips Hue*, *Facebook*) and (*Philips Hue*,

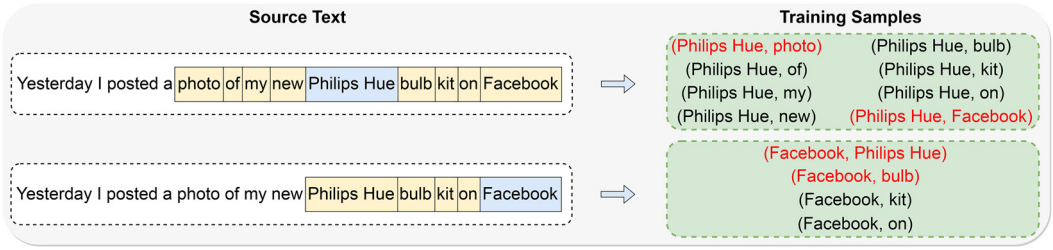


Fig. 3. Example of false PS (highlighted in red) generated through Skip-Gram.

*photo*) are labeled as PS, even though Facebook is a social network and Philips Hue is a solution for managing lights. Similarly, when using Facebook as the target channel, the words "Philips Hue," "bulb," "kit," and "on" are found in its context, generating the pair (*Facebook*, *bulb*) as a PS even though Facebook does not manage bulbs.

To categorize channels by functionality, we introduce the notion of *channel context*. In particular, a channel is contextual to a target channel if they share functionalities, yielding similar embeddings. Unfortunately, it is difficult to automatically determine the correct and relevant words from text descriptions, such as "post" for Facebook and "bulb" for Philips Hue. Manually defining these words is impractical due to the high number of channels and platform-specific variations. To address these issues, we propose *Channel2Vec*, a novel automated solution to define trustworthy positive and NS according to a threshold.

*Channel2Vec*. Figure 4 shows the architecture of Channel2Vec. It consists of four components: a module to query a virtual store, an embedding module, a similarity function paired with a threshold, and the Skip-Gram module. The concept behind the process is that channels with comparable functionalities should have similar descriptions on the digital marketplaces where they can be downloaded (such as the Google Play Store or App Store). As examples, Figure 5 shows a portion of the descriptions of the Facebook and Instagram channels available on the Google Play Store, with the similarities in text highlighted. We focus on a specific virtual store and gather information about the channels offered by a TAP, generating a collection of pairs $(n_{C_i}, d_{C_i})$, where $n_{C_i}$ represents the name of the TAP channel and $d_{C_i}$ represents the textual description of the channel's functionalities obtained from the virtual store.

To train the Skip-Gram model, we create *PS* with pairs of similar channels and *NS* with pairs of dissimilar channels. To achieve this goal, the descriptions are first processed through an embedding module, resulting in vector representations $W(d_{C_1}), W(d_{C_2}), \ldots, W(d_{C_{|V|}})$ containing channels' functional semantic information. Then, channel similarity $(n_{C_i}, n_{C_j})$ is evaluated by means of a similarity function *SF* and a threshold $\phi \in$ [-1,1]. If $SF(W(d_{C_i}), W(d_{C_j}))$ is greater or equal to $\phi$, then the label $y_{n_{C_i}, n_{C_j}}$ is set to 1 and the pair $(n_{C_i}, n_{C_j})$ is added to *PS*, otherwise, the label is set to 0 and the pair is added to *NS*. Higher $\phi$ values yield more reliable labels, but overly high thresholds may miss similar functionalities due to slight description differences, causing false negatives.

The Skip-Gram model predicts context words (channels) for a target word (channel) while modeling the vector representation of the latter. In particular, it learns an $N$-dimensional word embedding for $|V|$ channels, with $N$ constrained by the pre-trained LM. The model's input is a pair of words: one target channel $n_{C_t}$ and one context channel $n_{C_c}$. This pair feeds into an embedding layer of size $(|V| \times N)$, producing a dense embedding $(1 \times N)$ for both channels. A merge layer computes the dot product of the two embeddings, sending the result to a dense layer, producing prediction $y'_{n_{C_t}, n_{C_c}}$ as 1 or 0, depending on whether or not the channel pair is contextually relevant.

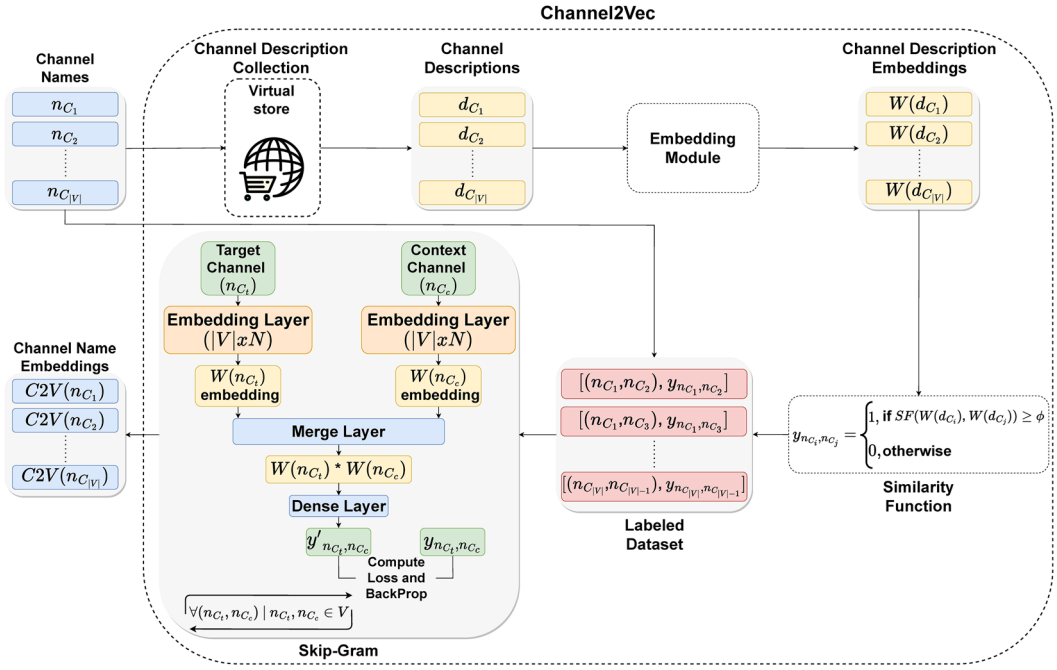


Fig. 4. Channel2Vec architecture.

**Instagram**

Instagram allows you to create and share your photos, stories, and videos with the friends and followers you care about. Connect with friends, share what you're up to, or see what's new from others all over the world. Explore our community where you can feel free to be yourself and share everything from your daily moments to life's highlights....

**Facebook**

Keeping up with friends is faster and easier than ever. Share updates and photos, engage with friends and Pages, and stay connected to communities important to you. Features on the Facebook app include:

* Connect with friends and family and meet new people on your social media network
* Set status updates & use Facebook emojis to help relay what's going on in your world
* Share photos, videos, and your favorite memories....

Fig. 5. Portions of Instagram and Facebook textual descriptions obtained from the Google Play Store with similar functionalities highlighted.

The Skip-Gram model calculates the sum of log probabilities for all pairs $(n_{C_t}, n_{C_c})$, resulting in binary cross-entropy loss function:

$$\mathcal{L} = -\frac{1}{|V|} \cdot \sum_{(n_{C_t}, n_{C_c}) \in PS \bigcup NS} \Big( \log P(y'_{n_{C_t}, n_{C_c}} = 1 \mid n_{C_t}, n_{C_c}, \Theta) \cdot y_{n_{C_t}, n_{C_c}} + \log P(y'_{n_{C_t}, n_{C_c}} = 0 \mid n_{C_t}, n_{C_c}, \Theta) \cdot (1 - y_{n_{C_t}, n_{C_c}}) \Big),$$

where $\Theta$ represents the model parameters. We want to maximize the likelihood of the PS, such as (Facebook, X (formerly known as Twitter)), while minimizing the likelihood of the NS, like (Facebook, Philips Hue). To this end, we leverage the sigmoid function to maximize

$$P(y'_{n_{C_t},n_{C_c}} = 1 \mid n_{C_t}, n_{C_c}, \Theta) = \sigma(v^T_{n_{C_c}} v_{n_{C_t}}) = \frac{1}{1 + e^{(-v^T_{n_{C_c}} v_{n_{C_t}})}} \quad \text{if } (n_{C_t}, n_{C_c}) \text{ is a positive sample, or}$$

$$P(y'_{n_{C_t},n_{C_c}} = 0 \mid n_{C_t}, n_{C_c}, \Theta) = 1 - P(y' = 1 \mid n_{C_t}, n_{C_c}, \Theta) = \frac{1}{1 + e^{(v^T_{n_{C_c}} v_{n_{C_t}})}} \quad \text{otherwise.}$$

Formally,

$$\Theta = \arg\max_{\Theta} \prod_{(n_{C_t},n_{C_c}) \in PS} P(y'_{n_{C_t},n_{C_c}} = 1 \mid n_{C_t}, n_{C_c}, \Theta) \cdot \prod_{(n_{C_t},n_{C_c}) \in NS} P(y'_{n_{C_t},n_{C_c}} = 0 \mid n_{C_t}, n_{C_c}, \Theta).$$

We compare the prediction with the actual label, compute the loss, and backpropagate the errors to adjust the weights in the embedding layer. This process is repeated cyclically on all pairs ($n_{C_t}$, $n_{C_c}$) for multiple epochs, yielding a set of channel embeddings $C2V \in \mathbb{R}^{|V| x N}$. In this way, the model can learn the contextually relevant channel pairs and generate similar embeddings for channels with similar functionalities. Moreover, in the inference phase, we only need the output embeddings and can discard the model.

*3.2.2 Training Strategies.* In contrast to traditional methods that only consider pre-trained model parameters, prompt learning-based tasks require consideration of both parameters and prompts. This makes the decision of which parameters to update a crucial part of the design process, impacting the model's performance. In the following, we summarize the main strategies for tuning pre-trained LM, highlighting those compatible with our proposal.

Existing training strategies differ in the parameters to be considered for the update [25]. We do not consider *Promptless Fine-Tuning* as it lacks prompts during pre-training and fine-tuning. Instead, *Tuning-free Prompting* enables the generation of texts by leveraging frozen pre-trained LM based on a prompt. It does not require parameter updates and is usually applied in zero-shot settings. Thus, we do not consider it since our proposal includes the training on a justification dataset. *Fixed-prompt LM Tuning* tunes the parameters of pre-trained LM and usually applies a discrete template to training and test samples. This method is suitable for discrete prompt learning, but we do not consider it for training our models as it does not handle soft prompts. Finally, *Fixed-LM Prompt Tuning* and *Prompt+LM Fine-tuning* strategies include prompt parameters updating, and thus they can be used with our hybrid prompt learning-based strategy [25]. Specifically, the former updates only the prompt parameters while freezing those of the pre-trained LM, whereas the latter updates both the prompt parameters and all or some of the parameters of the pre-trained LM.

## 4 Experimental Evaluation

This section describes the experiments conducted to assess the effectiveness of the proposed solution, including the user studies carried out to gather a dataset of user-defined justifications in the TAP domain. The study has been conducted by considering the IFTTT dataset introduced in [7], which contains rules labeled with specific harm by an NLP-based classification model. We evaluated the quality of the generated justifications using various metrics, which enabled us to assess their readability and compare them to the human-written justifications. Finally, we detail how pre-trained LM were customized for the generation task.

### 4.1 Evaluation Metrics

We adopt several automatic metrics widely used in natural language generation tasks, which have demonstrated a strong correlation with human ratings [10, 16, 39]. Specifically, we consider both embedding-based metrics, i.e., BLEURT [45], **Bidirectional and Auto-Regressive Transformers (BART)**Score [61], and MoverScore [65], as well as a word-overlap metric, i.e., **Metric for Evaluation of Translation with Explicit Ordering (METEOR)** [3]. These metrics measure the similarity between a generated text (candidate) with a "*gold standard*" (reference), which is a natural language text that has been annotated as a correct solution for a given task by humans. The higher the score, the closer the generated text is to the reference. Additionally, we employ a grammar-based metric, i.e., the **Coleman-Liau index (CLI)** readability test [12]. In contrast to the aforementioned metrics, CLI does not take references into consideration but focuses on a fundamental property of texts, i.e., the *readability*. Further details on the considered metrics are provided below:

- —*BLEURT*: A learned metric based on BERT, trained on the WMT Metrics Shared Task dataset[3] to capture semantic similarities [45]. To address domain drift, additional training data was created by perturbing Wikipedia sentences.
- —*BARTScore*: A metric proposed for evaluating generated text quality in NLP. It conceives evaluation as a text generation problem using the pre-trained BART model [61]. Specifically, it computes the weighted log probability of the target text given the source text using conditional probabilities from BART.
- —*MoverScore*: An automated evaluation metric for text generation systems that measures the semantic distance between system-generated texts and human references [65]. It utilizes contextualized representations to encode the texts and employs the Earth Mover's Distance as a powerful distance measure for semantic comparison.
- —*METEOR*: A metric based on the harmonic mean of precision and recall for unigrams [3]. It aims to identify the greatest number of words that can establish a connection between the reference and candidate text. It does this by calculating matches between individual words, considering Porter stemming and WordNet synonyms when there is no direct match. The result is penalized for candidates with correct words in the wrong order.
- —*CLI*: A readability test estimating the U.S. grade level needed to comprehend text [12]. It covers a wide range, from easy (<1) to highly academic (>16). A score of eight indicates an 8th-grade level in terms of comprehension. This test is specifically standardized for English and considers sentences and letters as key factors.

We use the first four metrics above to identify whether two sentences have the same meaning despite having different phrasing. In fact, a real-life threat scenario might be expressed in different ways. Therefore, metrics that only rely on lexical matching can result in inaccurate scores as they reward candidates who are similar to the references but do not convey the same meaning and penalize other paraphrases. In contrast, METEOR takes into account important factors, such as stemming and synonym matching, making it a more comprehensive evaluation tool. Metrics grounded in embeddings, which evaluate sentence meanings through the use of embeddings, capture relevant characteristics of justifications as words are represented dynamically by the words around them.

We use CLI to measure the ease with which the reader can comprehend the generated texts. This is an important aspect since the main users of TAPs tend to have limited technical knowledge in both security and privacy [53] and IoT technology [66], thus justifications must be clear and

[3]http://www.statmt.org/wmt20/metrics-task.html

straightforward, explaining potential harm without overwhelming the user with technical details that might confuse them.

### 4.2 Dataset Collection

We evaluated the proposed methodology using rules created through the IFTTT platform, which has been extensively studied in the literature due to its popularity [11, 47, 48]. In the following, we describe the process of creating the justification dataset for our case study.

The dataset on which we performed our study was initially collected in 2017 by Mi et al. through a scraping process [29]. The dataset comprised over 320,000 rules (also known as applets) gathered over a six-month period, with each applet consisting of: (1) a title (*Title*), (2) a description of its behavior (*Desc*), (3) the trigger component divided into its trigger (*TriggerTitle*) and relative channel (*TriggerChannelTitle*), and (4) the action component divided into its action (*ActionTitle*) and relative channel (*ActionChannelTitle*). To train and evaluate our LM, we considered the subset of 22,978 potentially risky applets, as identified by our harm identification algorithm [7], and randomly selected 3,709 applets ensuring a well-balanced distribution of triggers and actions, encompassing both IoT devices and online services. The algorithm categorizes applets into three harm classes: *Personal harm*, indicates potential user-induced data loss; *Physical harm*, represents physical damage to goods or individuals by third parties; and *Cybersecurity harm*, involves disrupting online services or distributing malware externally. The subset of applets comprised 2,044 instances for the Personal harm class, 727 instances for the Physical harm class, and 938 instances for the Cybersecurity harm class.

The statistics regarding the channel categories featured in the labeled dataset are presented in Table 1. The table includes information on the percentage of involvement in the rules for each channel category, as well as the percentage of applets whose triggers or actions are associated with a channel within the specified category. We can observe that most of the channels are for "Social networking, photo/video sharing," "Smarthome devices," and "Smartphones." Specifically, the most frequently used triggers for applets originate from "Social networking, photo/video sharing" and "Smartphones," while actions are predominantly associated with "Social networking, photo/video sharing" and "Smarthome devices."

The 3,709 applets underwent annotation by 18 independent annotators, who provided a justification as a real-life threat scenario for each of the harmful rules assigned to them. The procedure followed to collect these justifications has been organized as follows:

- A group of 15 annotators, comprised of both master's and bachelor's degree computer science students, provided justifications for 3,000 applets randomly selected. The authors only assessed the grammatical correctness of the justifications, manually correcting any misspellings or missing punctuation.
- The authors themselves annotated the remaining 709 applets.

### 4.3 Compared Methods

We performed a comprehensive comparative analysis, systematically examining our hybrid strategy alongside various baseline strategies, including discrete and hybrid prompt learning approaches. Discrete prompt learning baselines use manually-defined hard prompts proposed in Section 3.2.1, fine-tuning models conventionally. In contrast, hybrid prompt learning baselines use customized channel representations, integrating pre-trained representations for other rule information, diverging from Channel2Vec. It is important to note that while mapping channels as continuous tokens is recommended, using them solely as inputs in the prompt is infeasible due to behavioral differences among rules sharing the same channels as triggers and actions. For instance, rules such

Table 1. Statistics about the Channels Involved in the Applets of the Labeled Dataset

| | Channel category | % Channels | % Triggers | % Actions |
|---|---|---|---|---|
| IoT devices | Smarthome devices (e.g., light, thermostat) | 34.43% | 18.39% | 25.34% |
| | Smartphones (e.g., battery, NFC) | 31.54% | 25.67% | 15.07% |
| | Smarthome hub (e.g., Samsung SmartThings) | 11.65% | 4.07% | 8.17% |
| | Others | 2.86% | 2.45% | 0.4% |
| Online services | Social networking, photo/video sharing | 40.28% | 26.07% | 30.39% |
| | Cloud storage (e.g., Google Drive) | 13.78% | 2.97% | 11.16% |
| | SMS, instant messaging, VoIP | 20.54% | 15.02% | 8.14% |
| | Others | 6.66% | 5.37% | 1.32% |

as "*New tweet by you, turn off lights*" and "*New tweet by anyone in the area, turn on lights*" share the same channel composition but exhibit distinct behaviors, posing different risks. The former may be unintentionally triggered by the user, leading to lights turning off, while the latter could be maliciously exploited by an attacker to intentionally manipulate lights. Therefore, incorporating rule information and related harm in the prompt is essential to comprehensively understand rule behavior. Hence, we chose not to implement continuous prompt learning baselines.

*Discrete Baselines.* These models, serving as ablation studies, assess the impact of incorporating Channel2Vec for channel representation. Experiments treat all rule-related information, including channels, as discrete tokens. As a result, pre-trained model embeddings replace Channel2Vec-derived representations. The baselines include **Discrete Span-Infilling (DSI)** with a span-infilling prompt and **Discrete Question-Answering (DQA)** with a question-answering prompt.

*Hybrid Baselines.* These models employ hybrid prompt learning strategies as an alternative to Channel2Vec for embedding channels as continuous tokens. **Hybrid Skip-Gram (HSK)** uses the standard Skip-Gram with a specified context window size for generating positive and negative samples. **Sequential Tuning (ST)**, outlined in [22], involves two training stages: initial fine-tuning of continuous tokens with a frozen model, followed by updating both continuous tokens and model parameters. Both strategies maintain the remaining rule information as discrete tokens, organized in span-infilling (denoted as HSKSI and STSI, respectively) or question-answering (denoted as HSKQA and STQA, respectively) prompts, constituting a hybrid approach.

### 4.4 Implementation Details

In the following, we provide the implementation details of the proposed hybrid prompt learning-based strategy, denoted as **Hybrid Span-Infilling (HSI)** when employing the span-infilling prompt and **Hybrid Question-Answering (HQA)** when using the question-answering prompt, along with the details pertaining to the baseline models. All of them rely on pre-trained LM based on Transformer [49]. The dataset of applets was split into three sets: a training set with 2,967 applets, a validation set with 185 applets, and a test set with 557 applets.

*4.4.1 Implementation of the Proposed Hybrid Strategy.* As described in Section 3.2, the hybrid strategy relies on pre-trained LM to extend a prefix prompt and generate a justification. To accomplish this, we conducted experiments utilizing the subsequent models as the backbone for our hybrid strategy:

— *Phi [24]:* The Phi model is based on the Transformer architecture and employs multiple self-attention layers, which allow the model prioritize input text components during output generation. Trained on a diverse dataset, including synthetic textbooks and GPT-3.5-generated exercises, as well as carefully filtered web data, the Phi model gains access to a wide range of

information. The 1.5 version of the Phi model was employed for this work, consisting of 1.3 billion parameters and an embedding size of 2,048.

— *Pythia [4]:* The Pythia models adopt a decoder-only left-to-right Transformer architecture. Parameters, including the number of layers, hidden dimension size, and attention heads, are incrementally scaled based on well-established best practices. Unlike some earlier models, all Pythia models use fully dense attention. They also incorporate contemporary techniques like rotary positional encodings, enhancing performance over sinusoidal versions. Notably, the specific Pythia version used has 1 billion parameters and an embedding size of 2,048.

— *GPT-2 [42]:* The GPT-2 model is a large-scale LM designed to perform a wide range of NLP tasks without task-specific training data. Based on the Transformer architecture, the model comprises multiple layers featuring self-attention mechanisms and feed-forward neural networks. GPT-2 underwent pre-training on the WebText dataset with a causal LM objective. The large variant of GPT-2, featuring 774 million parameters and an embedding size of 1,280, is used in the experiments.

We chose these models to conduct a comprehensive comparison of similar architectures, avoiding challenges associated with data-hungry models. It is crucial to adopt models that strike a balance between effectiveness and the availability of training data. Larger models inherently require substantial datasets for optimal performance [22] and can strain computational resources. Moreover, fine-tuning large models can be time-consuming due to their complexity and increased parameter count.

We applied hard-soft prompts by augmenting applets with the span-infilling or question-answering prompts, considering the features *Title*, *Desc*, *TriggerTitle*, *ActionTitle*, and *Harm* as discrete tokens. The *TriggerChannelTitle* and *ActionChannelTitle* features (i.e., the channel used for the trigger and the action, respectively) were treated as continuous tokens and computed using the Channel2Vec embedding strategy. The list of IFTTT channels for embedding mapping was extracted from our proposed dataset [7], encompassing diverse categories, ranging from online services and content providers to smart home devices, totaling 435 channels. Channel descriptions were crawled on the Google Play Store app using the SELENIUM Python module. Sentence embeddings of descriptions were calculated by SentenceBERT [43], and cosine similarity was used for comparison. To study the effect of having more or fewer channels in the channel context of a target channel, we conducted several experiments using $\phi = \{0.75, 0.8, 0.85, 0.9\}$ as a threshold for constructing the labeled pairs (target channel, context channel). Following the specified embedding size $n$ for each model, we instantiated representations characterized by a dimensionality of $n$. We leveraged the TENSORFLOW Python library to build the deep learning architecture for the Skip-Gram model and trained it on the labeled pairs. Once the Skip-Gram model was trained, we extracted channel embeddings from the embedding layer and trained models using the *Prompt+LM Fine-tuning* strategy. Therefore, we initially encoded rules' channels through Channel2Vec and then fine-tuned soft prompts together with all parameters during training. We adopted the *negative log-likelihood* as a loss function, employed the *AdamW* algorithm as the optimizer, and set a *batch size* of 128 for GPT-2 and Pythia, and 60 for Phi. The *learning rate* was set to 1e-3, and the iterative process was terminated upon three consecutive instances of the loss function failing to decrease. Channel embeddings were placed as initial parameters to influence the representation processing of all tokens associated with the applet features. The model parameters were not frozen since the gap between prompt-tuning and fine-tuning disappears only with larger models (10 billion parameters) [21]. Thus, we did not apply the *Fixed-LM Prompt Tuning* strategy. Finally, we used the greedy decoding method and incorporated two special tokens, <*bos*> (begin-of-sequence token) and <*eos*> (end-of-sequence token), into the vocabulary. At each sequential step, the selection criterion for prediction involved identifying the word with the highest probability, which was then appended to the existing sequence to construct the subsequent input sequence for the ensuing step.

*4.4.2 Implementation of the Discrete Baselines.* For justification generation, DSI and DQA treat training and testing applet information as discrete tokens, enabling hard prompts. Thus, we trained using the *Fixed-prompt LM Tuning* strategy. Specifically, for both span-infilling and question-answering prompts, we considered the input text, i.e., applet information, identified harm, and prompt tokens, as discrete tokens for feeding pre-trained LM. The settings and backbones align with those delineated in Section 4.4.1.

*4.4.3 Implementation of the Hybrid Baselines.* For HSKSI and HSKQA strategies, we employed a context window size of three and utilized the dataset we proposed in [7] as the corpus, which consolidates information from each applet. Skip-Gram was then trained on labeled pairs, encompassing channels and applet-related keywords. Channel embeddings were extracted from the embedding layer, and models were trained using the *Prompt+LM Fine-tuning* strategy. In contrast, the STSI and STQA strategies combined *Fixed-LM Prompt Tuning* and *Prompt+LM Fine-tuning*. Initially, LM parameters were frozen, followed by prompt parameter optimization and fine-tuning of all model parameters. All strategies adhered to settings and backbones detailed in Section 4.4.1.

## 5 Results and Analysis

In this section, we will thoroughly assess the effectiveness of the justification generation methods outlined previously. This includes an examination of whether certain harms are more difficult to justify than others. Furthermore, to evaluate the impact of our hybrid prompt learning-based strategy, we conducted a manual analysis of the justifications generated for harmful applets involving unknown channels, i.e., channels not encountered during the LM training. Finally, we will present a comparative analysis, contrasting the outcomes obtained by our most effective model with those of two commercial LLMs.

### 5.1 Quantitative Analysis on Explainability and Readability

We analyze the impact of Channel2Vec on the generation of cohesive justifications by comparing our hybrid strategy with models using pre-trained embeddings for channel representations. Table 2 reports BLEURT, BARTScore, MoverScore, METEOR, and CLI scores obtained on the test set by both the discrete baselines and our hybrid strategy. The first four metrics assess the explainability of the generated justifications, measuring how well they align with the gold standards. Thus, high values are better. Conversely, CLI evaluates the readability of the generated justifications, determining their level of difficulty to read. In this case, low values are desirable. To assess the overall performance of the models and identify readability extremes, we compute CLI for all justifications and calculate their median, mean, maximum, and minimum. In the table, discrete baselines are denoted as $X_m$, whereas models leveraging the proposed hybrid strategy are denoted as $X_{m\text{-}\phi}$, where $X$ signifies the prompt strategy, $m$ corresponds to the backbone model, and $\phi$ is the Channel2Vec threshold.

Table 2 demonstrates the superiority of $HSI_{m\text{-}\phi}$ and $HQA_{m\text{-}\phi}$ strategies over their discrete counterparts trained with discrete tokens for channel name mapping. This emphasizes the efficacy of Channel2Vec in enhancing channel representations through interconnected relationships. The $HSI_{phi\text{-}0.85}$ strategy, incorporating the span-infilling prompt, Phi model, and a threshold $\phi$ of 0.85, yields the most favorable outcomes. Justifications generated by this model exhibit high explainability and satisfactory readability, equivalent to the $10th$ and $11th$ grades in the U.S. educational system. This indicates that the justifications can be easily understood by someone with a secondary school education. While justifications may occasionally be lengthy and repetitive (as indicated by the high Max value), they are generally straightforward and easily understood (as indicated by the low Min value). Notably, the threshold $\phi$ significantly influences model performance, with an improvement trend from 0.75 to 0.85 but a decline at 0.9. A too low threshold may include irrelevant channels

Table 2. Explainability and Readability of Models with Hybrid Strategy and Discrete Baselines

| | Explainability | | | | Readability | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **BLEURT** | **BARTScore** | **MoverScore** | **METEOR** | **Median** | **Mean** | **Max** | **Min** |
| $DSI_{phi}$ | 55 | 39 | 75 | 74 | 10.72 | 11.57 | 17.72 | 7.63 |
| $DQA_{phi}$ | 56 | 41 | 76 | 77 | 10.69 | 11.76 | 17.23 | 7.57 |
| $DSI_{pythia}$ | 58 | 42 | 77 | 79 | 10.51 | 11.34 | 16.79 | 6.52 |
| $DQA_{phytia}$ | 58 | 42 | 78 | 78 | 10.51 | 11.24 | 16.85 | 6.31 |
| $DSI_{gpt2}$ | 63 | 46 | 79 | 81 | 10.39 | 10.91 | 16.91 | 5.93 |
| $DQA_{gpt2}$ | 58 | 42 | 78 | 79 | 10.4 | 10.97 | 16.93 | 6.21 |
| $HSI_{phi\text{-}0.75}$ | 65 | 47 | 80 | 82 | 10.35 | 10.91 | 16.72 | 5.82 |
| $HSI_{phi\text{-}0.8}$ | 68 | 48 | 82 | 82 | 10.34 | 10.87 | 16.71 | 5.64 |
| $HSI_{phi\text{-}0.85}$ | **70** | **50** | **83** | **84** | **10.31** | 10.78 | **16.35** | 5.47 |
| $HSI_{phi\text{-}0.9}$ | 66 | 46 | 80 | 81 | 10.34 | 10.84 | 16.92 | 5.64 |
| $HSI_{pythia\text{-}0.75}$ | 66 | 46 | 80 | 81 | 10.51 | 11.25 | 16.75 | 5.76 |
| $HSI_{pythia\text{-}0.8}$ | 67 | 48 | 81 | 82 | 10.42 | 10.81 | 16.75 | 5.74 |
| $HSI_{pythia\text{-}0.85}$ | 67 | 49 | 82 | 82 | 10.35 | 10.61 | 16.31 | 5.12 |
| $HSI_{pythia\text{-}0.9}$ | 66 | 47 | 81 | 82 | 10.49 | 10.84 | 17.71 | 5.58 |
| $HSI_{gpt2\text{-}0.75}$ | 67 | 47 | 82 | 82 | 10.35 | 10.79 | 16.9 | 5.82 |
| $HSI_{gpt2\text{-}0.8}$ | 68 | 48 | 82 | 83 | 10.35 | 10.75 | 16.89 | 5.8 |
| $HSI_{gpt2\text{-}0.85}$ | **70** | 49 | **83** | 83 | 10.34 | 10.63 | 16.67 | 5.41 |
| $HSI_{gpt2\text{-}0.9}$ | 66 | 46 | 81 | 81 | 10.39 | 10.9 | 16.91 | 5.85 |
| $HQA_{phi\text{-}0.75}$ | 63 | 44 | 79 | 80 | 10.65 | 10.81 | 16.84 | 5.82 |
| $HQA_{phi\text{-}0.8}$ | 66 | 46 | 80 | 81 | 10.51 | 10.76 | 16.72 | 5.58 |
| $HQA_{phi\text{-}0.85}$ | 66 | 47 | 81 | 82 | 10.41 | 10.74 | 16.69 | 5.47 |
| $HQA_{phi\text{-}0.9}$ | 64 | 46 | 80 | 81 | 10.49 | 10.8 | 17.71 | 5.76 |
| $HQA_{pythia\text{-}0.75}$ | 64 | 45 | 80 | 79 | 10.34 | 10.76 | 16.94 | 5.76 |
| $HQA_{pythia\text{-}0.8}$ | 66 | 46 | 81 | 81 | 10.34 | 10.63 | 16.72 | 5.53 |
| $HQA_{pythia\text{-}0.85}$ | 66 | 47 | 81 | 83 | 10.34 | **10.56** | 16.72 | **4.83** |
| $HQA_{pythia\text{-}0.9}$ | 65 | 46 | 80 | 81 | 10.35 | 10.89 | 16.75 | 5.82 |
| $HQA_{gpt2\text{-}0.75}$ | 64 | 45 | 79 | 79 | 10.35 | 10.9 | 16.91 | 5.86 |
| $HQA_{gpt2\text{-}0.8}$ | 65 | 46 | 80 | 81 | 10.34 | 10.78 | 16.82 | 5.85 |
| $HQA_{gpt2\text{-}0.85}$ | 67 | 48 | 81 | 83 | 10.34 | 10.68 | 16.72 | 5.82 |
| $HQA_{gpt2\text{-}0.9}$ | 64 | 46 | 79 | 81 | 10.39 | 10.76 | 16.93 | 5.87 |

The values highlighted in bold represent the best scores achieved for each metric.

in the context of the target channel, complicating embeddings and rule context identification, while an excessively high threshold may result in too few related channels, challenging channel functionality mapping to the Skip-Gram model. Consequently, less favorable outcomes are observed with thresholds of 0.75 and 0.9.

Table 3 compares outcomes of the optimal-performing model using the proposed hybrid strategy ($HSI_{phi\text{-}0.85}$) with results from leading models employing discrete prompt learning ($DSI_{gpt2}$ and $DQA_{gpt2}$), and outcomes from HSK and ST hybrid strategies. Although HSK and ST strategies were applied across all models with prompt span-filling and question-answering, Table 3 reports only the models exhibiting superior performance (i.e., $HSKQA_{pythia}$ and $STSI_{phi}$) for brevity. Notably, all hybrid strategies outperform the use of discrete tokens for channel representation, emphasizing the suitability of continuous tokens for justifying risky automation rules. Among these, Channel2Vec yields the best results. Standard Skip-Gram pair construction is noisy and impedes channel relationship encoding. In contrast, the ST strategy associates channels specific to individual rules but lacks comprehensive functionality representation. Channel2Vec is the preferred choice as it encodes intrinsic relationships and reduces noise in continuous token computation.

To grasp the challenges of LM in justifying specific harmful rules, we analyzed the metrics for each class, focusing on high-performing models from prior experiments, i.e., $HSI_{phi\text{-}0.85}$ and $HSI_{gpt2\text{-}0.85}$. Table 4 displays BLEURT, BARTScore, MoverScore, and METEOR values for each

Table 3. Explainability and Readability of Best Models with Hybrid Strategy and Examined Baselines

| | Explainability | | | | Readability | | | |
|---|---|---|---|---|---|---|---|---|
| **Model** | **BLEURT** | **BARTScore** | **MoverScore** | **METEOR** | **Median** | **Mean** | **Max** | **Min** |
| $HSI_{phi\text{-}0.85}$ | **70** | **50** | **83** | **84** | **10.31** | **10.78** | **16.35** | **5.47** |
| $DSI_{gpt2}$ | 63 | 46 | 79 | 81 | 10.39 | 10.91 | 16.91 | 5.93 |
| $DQA_{gpt2}$ | 58 | 42 | 78 | 79 | 10.4 | 10.97 | 16.93 | 6.21 |
| $HSKQA_{pythia}$ | 65 | 46 | 81 | 81 | 10.39 | 10.85 | 16.87 | 5.92 |
| $STSI_{phi}$ | 67 | 47 | 81 | 81 | 10.38 | **10.78** | 16.68 | 5.58 |

The values highlighted in bold represent the best scores achieved for each metric.

Table 4. Explainability of Best Hybrid Prompt Learning-Based Models on Individual Classes of Harm

| | BLEURT | | | BARTScore | | | MoverScore | | | METEOR | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Model** | **PeH** | **PhH** | **CyH** | **PeH** | **PhH** | **CyH** | **PeH** | **PhH** | **CyH** | **PeH** | **PhH** | **CyH** |
| $HSI_{phi\text{-}0.85}$ | 69 | 56 | 74 | 49 | 36 | 55 | 79 | 74 | 85 | 81 | 76 | 88 |
| $HSI_{gpt2\text{-}0.85}$ | 63 | 42 | 75 | 47 | 38 | 55 | 80 | 73 | 85 | 83 | 72 | 88 |

PeH=Personal Harm; PhH=Physical Harm; CyH=Cybersecurity Harm.

harm class. Notably, rules in the Cybersecurity class are relatively easier to justify, with $HSI_{phi\text{-}0.85}$ and $HSI_{gpt2\text{-}0.85}$ achieving high metrics due to their focus on online services, which share similar functionalities. For instance, platforms like Facebook, X, and Instagram generally offer the same types of functionalities, such as publishing a post or updating profile pictures. Conversely, justifying harmful rules in the Physical class poses challenges for $HSI_{phi\text{-}0.85}$ and $HSI_{gpt2\text{-}0.85}$, as these rules involve physical devices, introducing complexity in disambiguating the context of execution. Managing diverse physical devices with multiple functionalities in a smart environment complicates the generation of consistent justifications. For example, smart lights can be turned on or off, or made to blink through the use of rules. While the LM may accurately identify devices, an erroneous focus on the wrong functionality can lead to inconsistent justifications. Harmful rules in the Personal class, dealing with self-inflicted harm across physical devices and online services, strike a balanced tradeoff between the aforementioned challenges, reflected in BLEURT, BARTScore, MoverScore, and METEOR values for $HSI_{phi\text{-}0.85}$ and $HSI_{gpt2\text{-}0.85}$.

## 5.2 Performances on Applets with Unknown Channels

To further underscore the superiority of our hybrid prompt learning-based strategy, we compare top-performing models trained using discrete and hybrid prompt learning strategies, i.e., $HSI_{phi\text{-}0.85}$, $HSI_{gpt2\text{-}0.85}$, $DSI_{gpt2}$, and $DQA_{gpt2}$, in scenarios where rules are defined through channels not seen during training. This poses a challenge for generating consistent justifications, as the models may struggle to grasp the context of rule execution involving unfamiliar channels. For correctness assessment, we manually examined justifications for 155 new harmful rules. Histograms in Figure 6 reveal that a significant proportion of errors stem from the generation of entirely inconsistent justifications, encompassing improper trigger and action events (represented by the third bar in each group). This can be attributed to the left-to-right generation mode of the models, whereby each word in a justification is influenced by its preceding words. As a result, if an erroneous trigger event is considered, it propagates inconsistent knowledge, leading to a failure to infer the action that caused harm. Notably, more errors occur when an incorrect action is generated compared to an erroneous trigger (represented by the first and second bars in each group). This is because

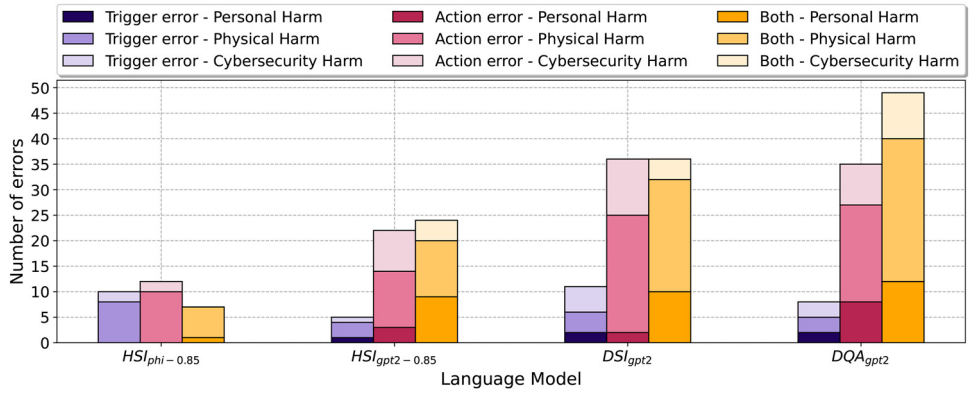


Fig. 6. Errors in justifications for rules with unknown channels by best discrete and hybrid strategies.

the trigger can be easily inferred from the rule information, even if it is maliciously exploited by an attacker, whereas the action refers to the event that could cause harm, which may not always be explicitly stated. Regarding the type of harm, Physical harms exhibit the highest error count, emphasizing the difficulty in justifying such rules. Additionally, Figure 6 indicates that $HSI_{phi\text{-}0.85}$ and $HSI_{gpt2\text{-}0.85}$ models perform best, showing the fewest errors.

To provide a practical overview of justification quality, we analyzed three harmful rules, each representing a different type of harm. Table 5 shows the characteristics of each rule, its associated harm, and the justifications generated by the examined models. Red sentences indicate events that do not align with the behavior of the corresponding rule, green sentences denote consistent descriptions, and blue emphasizes channels absent in the training set. We can observe that the models based on our hybrid strategy accurately identify discriminating features of rules, providing comprehensive justifications for the associated harm. These models excel in recognizing similarities among channels with akin functionalities, even in the presence of new channels, and adeptly capture the context of rule execution. Consequently, they are able to provide accurate justifications for harmful rules, ranging from general references like "user's device" to specific deductions, such as the $HSI_{phi\text{-}0.85}$ model foreseeing potential unexpected muting of the music if an attacker initiates malicious calls in the second harmful rule related to the Physical class.

Models mapping channel names to discrete tokens (i.e., $DSI_{gpt2}$ and $DQA_{gpt2}$), exhibit numerous inaccuracies in generated justifications. They struggle to identify triggering events or determining harmful actions, highlighting limited ability to disambiguate rule execution context. In practice, these models accurately capture a rule's behavior only in specific cases. For instance, in the first harmful rule, $DSI_{gpt2}$ accurately identifies potential harm from sensitive content dissemination on a social network but incorrectly attributes it to an attacker manipulating a user's sensor. Similarly, in the second rule, both $DSI_{gpt2}$ and $DQA_{gpt2}$ acknowledge harm from malicious calls but fail to identify the correct harm type. This highlights a significant issue with these models—their inability to comprehend execution context when faced with unseen channels during training.

### 5.3 Comparison with Commercial LLMs

This section compares our best model ($HSI_{phi\text{-}0.85}$) with two commercial LLMs (ChatGPT-3.5 and Claude) to highlight its strengths and limitations. Due to the precision needed in instructions for ChatGPT and Claude, we employed the span-infilling prompt, similar to $HSI_{phi\text{-}0.85}$, with the introductory text segment "Complete the text." The models were evaluated on the 557 rules of the test set. Due to potential disparities in justifications generated by the models and those defined

Table 5. Justifications Generated by Four Compared Models for Three Harmful Rules

| | | |
|---|---|---|
| INPUT | **Rule** | *Title*: **Tweet my daily Misfit activity summary**<br>*Desc*: **Keep me honest by tweeting my daily activity summary!**<br>*TriggerChannelTitle*: **Misfit** — *ActionChannelTitle*: **Twitter**<br>*TriggerTitle*: **Daily activity summary** — *ActionTitle*: **Post a tweet** |
| | **Harm** | Personal |
| OUTPUT | $HSI_{phi\text{-}0.85}$ | If the user **gets negative results in his/her daily activity summary**, they will automatically be **published on the social network**, causing embarrassing situations. |
| | $HSI_{gpt2\text{-}0.85}$ | If the **daily activity summary results achieved are below expectations**, they will automatically be **posted on the social network**, causing embarrassing situations. |
| | $DSI_{gpt2}$ | An attacker **could pass in front of the sensor multiple times** and continuously cause the **posting of sensitive content on the social network**. |
| | $DQA_{gpt2}$ | An attacker **could pass in front of the sensor multiple times** and continuously cause the **sending of messages to the user**. |
| INPUT | **Rule** | *Title*: **If the phone rings mute the music**<br>*Desc*: **Mutes playback on all Musaic players when your Android phone rings**<br>*TriggerChannelTitle*: **Android Phone Call** — *ActionChannelTitle*: **Musaic**<br>*TriggerTitle*: **Any phone call missed** — *ActionTitle*: **Mute** |
| | **Harm** | Physical |
| OUTPUT | $HSI_{phi\text{-}0.85}$ | An attacker **could make spam calls** in such a way as **to mute the music**. |
| | $HSI_{gpt2\text{-}0.85}$ | An attacker **could repeatedly make spam calls** in order **to manipulate the status of a user's device**. |
| | $DSI_{gpt2}$ | An attacker **could make repeated calls** to enable **the creation of a smoke alarm**. |
| | $DQA_{gpt2}$ | An attacker **could make repeated calls** to allow **spam messages to be sent to the user**. |
| INPUT | **Rule** | *Title*: **An eye on unread incoming e-mails**<br>*Desc*: **This applet searches for new unread emails in my Gmail inbox and sends me a notification**<br>*TriggerChannelTitle*: **Gmail** — *ActionChannelTitle*: **Stack**<br>*TriggerTitle*: **New email in inbox from search** — *ActionTitle*: **Send a notification** |
| | **Harm** | Cybersecurity |
| OUTPUT | $HSI_{phi\text{-}0.85}$ | An attacker **could send malicious emails** to allow **repeated notifications to be sent to the user's device**. |
| | $HQA_{gpt2\text{-}0.85}$ | An attacker **could send several emails** in order **to fill the user's device with spam notifications**. |
| | $DSI_{gpt2}$ | An attacker **could send malicious SMS** in such a way as **to automatically send spam emails**. |
| | $DQA_{gpt2}$ | An attacker **could pass in front of the sensor multiple times** and continuously cause **spam emails to be sent to the user's device**. |

Blue words highlight channels not included in the training set. Green sentences indicate cases where the execution context of the trigger/action is correctly identified, while red sentences show cases where a wrong event is generated.

by the annotators, we manually verified their correctness without considering the previously introduced metrics. From the manual verification, we observed that ChatGPT generated inaccurate justifications for 221 rules, Claude for 228, and $HSI_{phi\text{-}0.85}$ for 74.

ChatGPT demonstrated superior justification quality, proficiently generating technically detailed and accessible versions. This proficiency stems from the reinforcement learning from human feedback approach and diverse expert inclusion in training. However, ChatGPT often focused on specific rule aspects, generating implausible scenarios or deviating from overall trigger-action-dependent behavior. For example, in a rule "*IF you enter an area THEN blink lights*" and description "*Gives the spouse a chance to get any liaisons out and/or to make you a nice drink! Cheers*," ChatGPT's justification emphasized semantics from the description (a joke) rather than considering potential inconveniences, like lights blinking during bedtime. Indeed the justification contains sentences like "*[...] the statement 'Gives spouse a chance to get any liaisons out' implies a potential breach of trust within a relationship. Additionally, the phrase 'make you a nice drink' could be interpreted as promoting or enabling excessive alcohol consumption, which can be harmful to one's health and well-being [...].*" Conversely, Claude generated suboptimal justifications, occasionally reporting

an inability due to insufficient context or "*a discomfort in suggesting how connected devices could potentially cause harm.*" This precaution is likely a deliberate safeguard employed by the researchers to discourage exploitation by attackers for orchestrating malicious activities.

Unlike ChatGPT-3.5 and Claude, the $HSI_{phi\text{-}0.85}$ model often discards contextual relevance in input rules, resulting in justifications incongruent with the trigger or action of the specified rule. Nevertheless, it consistently produces accurate justifications for the majority of rules. This highlights the ability of a smaller model (Phi 1.5, 1.3b parameters), refined through fine-tuning on a task-specific dataset against larger-scale models, enabling a controlled generative process.

## 6 Conclusion and Future Work

We have introduced a hybrid prompt learning strategy for generating post hoc natural language justifications of security risks in trigger-action rules. The generated justifications are framed in real-life threat scenarios and aim to increase awareness among non-technical users. The approach uses a combination of hard and soft prompts derived from automation rule components, and incorporates a novel strategy named Channel2Vec that generates continuous tokens, capturing domain-specific relationships between words. We found that the justifications generated using the proposed models are highly readable and are deemed to accurately describe the security risks.

In the future, we intend to investigate the application of self-rationalization techniques [28], which enable LM to both identify the harm associated with a rule and generate the corresponding justification. Through the evaluation of the consistency between the generated justifications and the identified harms, LM could identify areas where their understanding may be flawed and adjust the output accordingly. Moreover, we intend to apply the proposed Channel2Vec strategy in other application domains, such as for justifying song recommendations [64], where continuous tokens (computed by exploiting music platforms like Spotify) could emphasize the domain-specific relationship between elements, such as singer or band names, providing a more nuanced understanding than discrete tokens alone.

## References

[1] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. 2019. A convergence theory for deep learning via over-parameterization. In *Proceedings of the International Conference on Machine Learning* (*ICML '19)*. 242–252.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The internet of things: A survey. *Comput. Netw.* 54, 15 (2010), 2787–2805.

[3] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (*ACL StatMT '07)*. 65–72.

[4] Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, Usvsn Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *Proceedings of the International Conference on Machine Learning* (*ICML '23)*. 2397–2430.

[5] Or Biran and Courtenay Cotton. 2017. Explanation and justification in machine learning: A survey. In *Proceedings of the IJCAI-17 Workshop on Explainable AI (XAI '17)*, Vol. 8. 8–13.

[6] Bernardo Breve, Gaetano Cimino, and Vincenzo Deufemia. 2022. Towards explainable security for ECA rules. In *Proceedings of the EMPATHY '21 Workshop*. 33–37.

[7] Bernardo Breve, Gaetano Cimino, and Vincenzo Deufemia. 2023. Identifying security and privacy violation rules in trigger-action IoT platforms with NLP models. *IEEE IoT J.* 10, 6 (2023), 5607–5622.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford,

Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the Advances in Neural Information Processing Systems,* Vol. 33. 1877–1901.

[9] Z. Berkay Celik, Patrick McDaniel, and Gang Tan. 2018. Soteria: Automated IoT safety and security analysis. In *Proceedings of the 2018 USENIX Annual Technical Conference (USENIX ATC '18)*. 147–158.

[10] Miruna-Adriana Clinciu, Arash Eshghi, and Helen F. Hastie. 2021. A study of automatic metrics for the evaluation of natural language explanations. In *Proceedings of the European Chapter of the Association for Computational Linguistics* (*EACL '21)*. 2376–2387.

[11] Camille Cobb, Milijana Surbatovich, Anna Kawakami, Mahmood Sharif, Lujo Bauer, Anupam Das, and Limin Jia. 2020. How risky are real users' IFTTT applets? In *Proceedings of the 16th Symposium on Usable Privacy and Security* (*SOUPS '20)*. 505–529.

[12] Meri Coleman and Ta Lin Liau. 1975. A computer readability formula designed for machine scoring. *J. Appl. Soc. Psychol.* 60, 2 (1975), 283.

[13] Fulvio Corno, Luigi De Russis, and Alberto Monge Roffarello. 2019. RecRules: Recommending IF-THEN rules for end-user development. *ACM Trans. Intell. Syst. Technol.* 10, 5 (2019), 1–27.

[14] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. 2018. Automatic generation of natural language explanations. In *Companion Proceedings of the 23rd International Conference on Intelligent User Interfaces* (*IUI '18)*. 1–2.

[15] Giuseppe Desolda, Carmelo Ardito, and Maristella Matera. 2017. Empowering end users to customize their smart environments: Model, composition paradigms, and domain-specific tools. *ACM Trans. Comput.-Hum. Interact.* 24, 2 (2017), 1–52.

[16] Desmond Elliott and Frank Keller. 2014. Comparing automatic evaluation measures for image description. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics* (*ACL '14)*. 452–457.

[17] Chenglong Fu, Qiang Zeng, and Xiaojiang Du. 2021. HAWatcher:Semantics-aware anomaly detection for appified smart homes. In *Proceedings of the 30th USENIX Security Symposium* (*USENIX Security '21)*. 4223–4240.

[18] Matt Gardner, Jonathan Berant, Hannaneh Hajishirzi, Alon Talmor, and Sewon Min. 2019. Question answering is a format; When is it useful? arXiv:1909.11291. Retrieved from https://arxiv.org/abs/1909.11291

[19] Giuseppe Ghiani, Marco Manca, Fabio Paterno, and Carmen Santoro. 2017. Personalization of context-dependent applications through trigger-action rules. *ACM Trans. Comput.-Hum. Interact.* 24, 2 (2017), 1–33.

[20] Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2022. PTR: Prompt tuning with rules for text classification. *AI Open* 3 (2022), 182–192.

[21] Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the Empirical Methods in Natural Language Processing* (*EMNLP '21)*. 3045–3059.

[22] Lei Li, Yongfeng Zhang, and Li Chen. 2023b. Personalized prompt learning for explainable recommendation. *ACM Trans. Inf. Syst.* 41, 4 (2023), 1–26.

[23] Xiang Lisa Li and Percy Liang. 2021. Prefix-Tuning: Optimizing continuous prompts for generation. In *Proceedings of the Findings of the Association for Computational Linguistics* (*ACL/IJCNLP '21)*. 4582–4597.

[24] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023a. Textbooks are all you need II: phi-1.5 technical report. arXiv:2309.05463. Retrieved from https://arxiv.org/abs/2309.05463

[25] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Comput. Surveys* 55, 9 (2023), 1–35.

[26] Xuanyu Liu, Xiao Fu, Xiaojiang Du, Bin Luo, and Mohsen Guizani. 2022. Machine learning based non-intrusive digital forensic service for smart homes. *IEEE Trans. Netw. Serv. Manag.* 20, 2 (2022), 945–960.

[27] Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023b. GPT understands, too. *AI Open* (2023).

[28] Ana Marasovic, Iz Beltagy, Doug Downey, and Matthew E. Peters. 2022. Few-shot self-rationalization with natural language prompts. In *Proceedings of the North American Chapter of the Association for Computational Linguistics* (*NAACL '22)*. 410–424.

[29] Xianghang Mi, Feng Qian, Ying Zhang, and XiaoFeng Wang. 2017. An empirical characterization of IFTTT: ecosystem, usage, and performance. In *Proceedings of the Internet Measurement Conference* (*IMC '17)*. 398–404.

[30] Sabrina J. Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y. Lee, Benoît Sagot, and Samson Tan. 2021. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. arXiv:2112.10508. Retrieved from https://arxiv.org/abs/2112.10508

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. arXiv:1301.3781. Retrieved from https://arxiv.org/abs/1301.3781

[32] Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2021. Generating post hoc review-based natural language justifications for recommender systems. *User Model. User-Adapt. Interact.* 31, 3 (2021), 629–673.

[33] Dang Tu Nguyen, Chengyu Song, Zhiyun Qian, Srikanth V. Krishnamurthy, Edward J. M. Colbert, and Patrick McDaniel. 2018. IotSan: Fortifying the safety of IoT systems. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (*CoNEXT '18*). 191–203.

[34] Steven Ovadia. 2014. Automate the internet with "if this then that" (IFTTT). *Behav. Soc. Sci. Librar.* 33, 4 (2014), 208–211.

[35] Federica Paci, Davide Bianchin, Elisa Quintarelli, and Nicola Zannone. 2020. IFTTT privacy checker. In *Proceedings of the International Workshop on Emerging Technologies for Authorization and Authentication* (*ETAA '20*). 90–107.

[36] Dong Huk Park, Lisa Anne Hendricks, Zeynep Akata, Anna Rohrbach, Bernt Schiele, Trevor Darrell, and Marcus Rohrbach. 2018. Multimodal explanations: Justifying decisions and pointing to the evidence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR '18*). 8779–8788.

[37] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics* (*NAACL '18*). 2227–2237.

[38] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (*EMNLP-IJCNLP '19*). 2463–2473.

[39] Atharva Phatak, David W Savage, Robert Ohle, Jonathan Smith, and Vijay Mago. 2022. Medical text simplification using reinforcement learning (TESLEA): Deep learning–based text simplification approach. *JMIR Med. Inf.* 10, 11 (2022), e38095.

[40] Wolter Pieters. 2011. Explanation and trust: what to tell the user in security and AI? *Ethics Inf. Technol.* 13, 1 (2011), 53–64.

[41] Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (*NAACL-HLT '21*). 5203–5212.

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[43] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing* (*EMNLP-IJCNLP '19*). 3982–3992.

[44] Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the European Chapter of the Association for Computational Linguistics* (*EACL '21*). 255–269.

[45] Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In Proceedings of the Association for Computational Linguistics (*ACL '20*). 7881–7892.

[46] Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the Empirical Methods in Natural Language Processing* (*EMNLP '21*). 7699–7715.

[47] Milijana Surbatovich, Jassim Aljuraidan, Lujo Bauer, Anupam Das, and Limin Jia. 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* (*WWW '17*). 1501–1510.

[48] Blase Ur, Melwyn Pak Yong Ho, Stephen Brawner, Jiyun Lee, Sarah Mennicken, Noah Picard, Diane Schulze, and Michael L Littman. 2016. Trigger-action programming in the wild: An analysis of 200,000 IFTTT recipes. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (*CHI '16*). 3227–3231.

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in Neural Information Processing Systems* (*NIPS '17*). 6000–6010.

[50] Luca Vigano and Daniele Magazzeni. 2020. Explainable security. In *Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 293–300.

[51] Guangjing Wang, Hanqing Guo, Anran Li, Xiaorui Liu, and Qiben Yan. 2023a. Federated IoT interaction vulnerability analysis. In *Proceedings of the IEEE 39th International Conference on Data Engineering* (*ICDE '23*). 1517–1530.

[52] Guangjing Wang, Nikolay Ivanov, Bocheng Chen, Qi Wang, ThanhVu Nguyen, and Qiben Yan. 2023b. Graph learning for interactive threat detection in heterogeneous smart home rule data. *Proc. ACM Manag. Data* 1, 1 (2023), 1–27.

[53] Qi Wang, Pubali Datta, Wei Yang, Si Liu, Adam Bates, and Carl A. Gunter. 2019. Charting the attack surface of trigger-action IoT platforms. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security* (*CCS '19*). 1439–1453.

[54] Qi Wang, Wajih Ul Hassan, Adam Bates, and Carl Gunter. 2018. Fear and logging in the internet of things. In *Proceedings of the Network and Distributed Systems Symposium*. 1–16.

[55] Karl Weiss, Taghi Khoshgoftaar, and DingDing Wang. 2016. A survey of transfer learning. *J Big Data* 3, 1 (2016), 1–40.

[56] Ding Xiao, Qianyu Wang, Ming Cai, Zhaohui Zhu, and Weiming Zhao. 2019. A3ID: An automatic and interpretable implicit interference detection method for smart home via knowledge graph. *IEEE IoT J.* 7, 3 (2019), 2197–2211.

[57] Zhouhang Xie, Sameer Singh, Julian J. McAuley, and Bodhisattwa Prasad Majumder. 2023. Factual and informative review generation for explainable recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence* (*AAAI '23)*. Brian Williams, Yiling Chen, and Jennifer Neville (Eds.). 13816–13824.

[58] Yuhang Yao, Mohammad Mahdi Kamani, Zhongwei Cheng, Lin Chen, Carlee Joe-Wong, and Tianqiang Liu. 2023. FedRule: Federated rule recommendation system with graph neural networks. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation* (*IoTDI '23)*. 197–208.

[59] Raciel Yera, Ahmad A. Alzahrani, and Luis Martinez. 2022. Exploring post-hoc agnostic models for explainable cooking recipe recommendations. *Knowl. Based Syst.* 251 (2022), 109216.

[60] Hao Yuan, Haiyang Yu, Jie Wang, Kang Li, and Shuiwang Ji. 2021b. On explainability of graph neural networks via subgraph explorations. In *Proceedings of the International Conference on Machine Learning* (*ICML '21)*. 12241–12252.

[61] Weizhe Yuan, Graham Neubig, and Pengfei Liu. 2021a. BARTScore: Evaluating generated text as text generation. In *Proceedings of the Advances in Neural Information Processing Systems* (*NeurIPS '21)*. 27263–27277.

[62] Imam Nur Bani Yusuf, Lingxiao Jiang, and David Lo. 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* (*ICPC '22)*. 99–110.

[63] Yongfeng Zhang and Xu Chen. 2020. Explainable recommendation: A survey and new perspectives. *Found. Trends Inf. Retr.* 14, 1 (2020), 1–101.

[64] Guoshuai Zhao, Hao Fu, Ruihua Song, Tetsuya Sakai, Zhongxia Chen, Xing Xie, and Xueming Qian. 2019a. Personalized reason generation for explainable song recommendation. *ACM Trans. Intell. Syst. Technol.* 10, 4 (2019), 1–21.

[65] Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019b. MoverScore: Text generation evaluating with contextualized embeddings and Earth mover distance. In *Proceedings of the Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing* (*EMNLP-IJCNLP '19)*. 563–578.

[66] Serena Zheng, Noah Apthorpe, Marshini Chetty, and Nick Feamster. 2018. User perceptions of smart home IoT privacy. *Proc. ACM Hum.-Comput. Interact.* 2 (2018), 1–20.

Received 9 March 2023; revised 13 February 2024; accepted 29 May 2024