

# Towards a Write Once Run Anywhere Approach in End-User IoT Development

Ekene Attoh<sup>1</sup> <sup>a</sup>, Beat Signer<sup>1</sup> <sup>b</sup>

<sup>1</sup>Web & Information Systems Engineering Lab, Vrije Universiteit Brussel, Pleinlaan 2, Brussels, Belgium  
{eattoh, bsigner}@vub.be

**Keywords:** Cross-platform IoT, natural language processing, end-user authoring, semantic interoperability

**Abstract:** With the rise of popular task automation or IoT platforms such as *If This Then That (IFTTT)*, users can define rules to enable interactions between smart devices in their environment and thereby improve their daily lives. However, the rules authored via these platforms are usually tied to the platforms and sometimes even to specific devices for which they have been defined. Therefore, when a user wishes to move to a different environment controlled by a different platform and/or devices, they need to recreate their rules for the new environment. The rise in the number of smart devices further adds to the complexity of rule authoring since users will have to navigate an ever-changing landscape of IoT devices. In order to address this problem, we need human-computer interaction that works across the boundaries of specific IoT platforms and devices. A step towards this human-computer interaction across platforms and devices is the introduction of a high-level semantic model for end-user IoT development, enabling users to create rules at a higher level of abstraction. However, many users who are used to the rule representation in their favourite tool might be unwilling to learn and adapt to a new representation. We present a method for translating proprietary rules to a high-level semantic model by using natural language processing techniques. Our translation enables users to work with their familiar rule representation language and tool, but then apply their rules across different IoT platforms and devices.


## 1 INTRODUCTION


The interoperability issue between devices of different brands in the domain of the Internet of Things (IoT) is omnipresent, a main reason being the unwillingness of major device manufacturers to make their devices to interoperate with their competitors' devices (Longo et al., 2022). (Noura et al., 2019) studied different categories interoperability issues in state-of-the-art IoT solutions and found that most IoT solutions do not support *cross-platform* and *cross-domain* interoperability. If supported, these categories can enable IoT users to exploit different IoT services independently of the platform (e.g. Apple or Samsung) or domain (e.g. health or mobility). End-user development enables users to develop and adapt systems in line with their background and skills (Barricelli et al., 2019) and has been suggested to give users control over IoT solutions.

Various tools have been proposed to support end-user IoT development (Desolda et al., 2016; Coutaz

and Crowley, 2016; Ospan et al., 2018). (Markopoulos et al., 2017) noted that the most common programmatic end-user control of IoT applications is through specifying rules. Previous studies (Cabitzza et al., 2017; Ur et al., 2016) have further shown that a rule-based approach is easily understandable and enables end users to create their own programs. (Li et al., 2017) identified that although the popular IFTTT<sup>1</sup> IoT task automation platform enables users to create rules across various devices and services, not every kind of device or service is supported.

(Corno et al., 2021) stated that most end-user development platforms adopt a vendor-centric abstraction, thus requiring that every online service needs to be programmed in a specific way. They argued that this poses interoperability challenges since users need to know any technological details to execute the intended behaviours beforehand. This approach is inadequate in future IoT environments like smart cities, as *things* will not always be known a priori but might dynamically appear and disappear (Corno et al., 2021). As stated by (Attoh and Signer, 2021), tackling in-

<sup>a</sup>  <https://orcid.org/0000-0002-8590-1005>

<sup>b</sup>  <https://orcid.org/0000-0001-9916-0837>

<sup>1</sup><https://ifttt.com>

interoperability issues ensures that users are no longer locked-in to vendor-specific tools or platforms.

We start with an overview of related work addressing interoperability in IoT environments. Our NLP-based solution for automatic rule translation is presented in Section 3 and the performance of three translation algorithms is discussed in Section 4. Users are enabled to work with their familiar rule representation language and tool, and at the same time can apply their rules across different IoT platforms and devices. After an initial user evaluation of our *write once, run anywhere* approach for end-user IoT development presented in Section 5, we provide some conclusions and outlook for future work.

## 2 RELATED WORK

To address the interoperability issue across different IoT solutions, (Li et al., 2017) presented a solution allowing users to author rules for their IoT devices by demonstrating interactions between smart devices using their mobile phones. The solution consists of an Android application enabling users to create automation scripts composed by recording the actions users perform on the mobile application of their smart devices. The scripts can then be triggered to perform the actions, using a source from another application such as a notification from a motion sensor application. Note that since the solution was designed to work on the Android operating system, the created automations will only be usable on Android devices. This implies that a user would lose all their automations if they were, for example, going to switch to an iOS device.

(Corno et al., 2019) attempted to address the interoperability issue by introducing the *EUPont* (End User Programming Ontology) high-level semantic model for end-user IoT development. With *EUPont*, users no longer need to create rules for specific devices or services, but they can define abstract rules such that any device or service able to perform the required action can be used to execute those rules. For instance, a specific IFTTT rule like “*If my smart sensor X detects that I am home and the outside temperature is less than 10 degrees, then turn on my smart heater H*” would have to be recreated if a user switches to a smart sensor “*Y*” or smart heater “*I*”. In *EUPont*, the rule can be transformed to “*If I am in an indoor place and the outside temperature is less than 10 degrees, then start heating the indoor place*”. Using any platform understanding the *EUPont* ontology, a user’s rules can be executed on any device that can detect their presence and also heat their environment.

Another factor contributing to the problem of interoperability in IoT is an issue present in end-user IoT development. As more technologies are supported by platforms such as IFTTT, the design space also grows and it becomes more difficult for users to discover rules and their related functionality (Corno et al., 2020b). This increased complexity makes interoperability more difficult since rules with similar functionality can be created in different ways. The use of recommendations in end-user development tools has been proposed to address this issue, similar as for the development of software artefacts (Krüger, 2018; Nguyen et al., 2016). However, the opportunities for recommendations have not yet been consistently explored to support end-user development, but rather focus on helping professional developers (Corno et al., 2020b). Therefore, the *TAPrec* end-user development platform enabling the composition of trigger-action rules based on dynamic recommendations has been introduced. At composition time, it suggests new rules to be used or actions, which are based on the rule’s final purpose such as illuminating a place rather than details like device brands and manufacturers, for completing a rule.

(Mattioli and Paternò, 2021) proposed a solution that suggests relevant triggers, operators and actions to a user during rule composition. The system provides both, step-by-step and full-rule recommendations and a user is either recommended components to complete their rule or the system suggests a complete rule. (Jeong et al., 2019) introduced a framework to analyse the usage logs of devices in an IoT context and make rule recommendations to users based on the analysis of their device usage patterns.

*InstructableCrowd* (Huang et al., 2016) is a crowd-sourcing system enabling users to create IF-THEN rules based on their needs. Users can describe their problems (e.g. often being late for a meeting) via a smartphone user interface to *crowdworkers*, and the *crowdworkers* create rules addressing a user’s needs and send them back to their phone. *HeyTAP* (Corno et al., 2020a) supports users in describing the desired behaviour of their smart devices through conversations (text or voice) and getting rule recommendations materialising their stated intentions.

The *situation* concept proposed by (Trullemans et al., 2017) and implemented in the Context Modelling Toolkit (CMT) is another means to tackle the complexity of authoring rules with similar functionality. They proposed that the *trigger* side of a rule can lead to the definition of a reusable *situation* rather than just triggering an action. This situation can then be used on the *trigger* side of a new rule definition, eliminating the need for users to understand all the

low-level details since they can also use situations in the definition of their automations.

Although related work proposed solutions to address IoT interoperability issues, to the best of our knowledge they are only focusing on the creation of *new* rules by users in novel ways and using *new* systems to bridge the interoperability gap rather than enabling users to retain their current tools and methods, while still being able to benefit from solutions offering cross-platform interoperability (Attoh and Signer, 2021). Based on our analysis of related work, we identified two major problems to be addressed:

**Loss of Tooling Choice:** As mentioned before, various solutions have been put forward to bridge the cross-platform gap (Li et al., 2017; Corno et al., 2019), but they also propose the use of new tools and languages. This means that users need to learn to use new tools and languages for creating rules that work across different platforms.

**Rule Authoring Complexity:** Related work further shows that due to the rise in the number of smart devices, the discoverability of rules and their related functionality becomes more complex (Corno et al., 2020b). Therefore, users do not only need to use new tools and rule description languages to benefit from cross-platform interoperability solutions, but they also have to navigate an ever-changing landscape of IoT devices and services while authoring their rules. This additional complexity may not only create an entry barrier for new users, but also increase a user’s time needed to create their desired automation. We analysed the IFTTT user recipes (rules) from the May 2017 dataset of Mi et al. (Mi et al., 2017) and found that out of the total 279 828 user recipes, there were 863 duplicate triggers (number of triggers that were used more than once) and 502 duplicate actions (number of actions that were used more than once). The identification of these triggers and actions—as well as understanding their functionality—can become more difficult for lesser-known triggers and actions resulting in unmanageable complexity for most users and in particular for non-expert users.

### 3 RULE TRANSLATION

We propose a Natural Language Processing (NLP) approach for automatically translating proprietary end-user rules to the EUPont high-level abstraction by Corno et al. (Corno et al., 2019). This method provides end users with a *Write Once, Run Anywhere* paradigm where they can retain the authoring tool and language description of their choice but have the flexibility to use their rules across different platforms, en-

abling IoT interoperability on the application layer. A user simply has to write their rule as they would normally do and have it translated to an equivalent EUPont representation. For instance, let us assume that a user has previously composed the IFTTT rule “*If AC brand X is turned off, then activate my camera brand Y*” for their smart home. They now find themselves on vacation in a smart environment which uses an air conditioner (AC) of brand Z and a camera of brand C. With existing solutions, the user needs to create a new rule “*If AC brand Z is turned off, then activate my camera brand C*” to have the same experience in their vacation environment as they would enjoy at their home. We rather propose a solution where a rule can be automatically translated to the EUPont generalisation “*If device turned off, then connect to device*”. Just as the JSON<sup>2</sup> format is generic such that most (modern) language compilers and interpreters are equipped with JSON parsers, the intention behind the EUPont representation is that IoT platforms with the EUPont “*runtime*” might be able to work with the representation. Therefore, a proprietary rule (e.g. IFTTT rule) has to be written only *once* and can be translated to the EUPont representation to be used across different platforms. This means that an EUPont-powered platform can make it possible for any device which might be triggered off to be used as the *trigger* of the rule. For the rule’s action, a camera can be mapped to the high-level action “*Connect to device*”, which can then be triggered when the rule is executed. A user is therefore not limited to using devices of brand X or Y in order to take advantage of their already composed rule. Note that our rule translation has been described in detail in a technical report (Attoh and Signer, 2023a).

An overview of our IoT rule translation approach is shown in Figure 1. A user creates (proprietary) IoT rules, which can be seen as the *Write Once* part, using any platform of their choice (e.g. IoT Platform A or IoT Platform B). These platforms have access to our translation approach described later, and the Translation Module then converts the created rules to the high-level EUPont representation.

As stated previously, a user would need to duplicate and further customise a rule authored to work on a specific platform to use that rule on a different platform, given that each platform stores its users’ data locally. In order to address this issue, we introduce the use of Solid Pods to store the automatically translated high-level rules (Sambra et al., 2016). Solid aims to provide data independence as well as simple yet powerful data management mechanisms, where applications no longer store their data themselves, but request

<sup>2</sup><https://www.json.org>

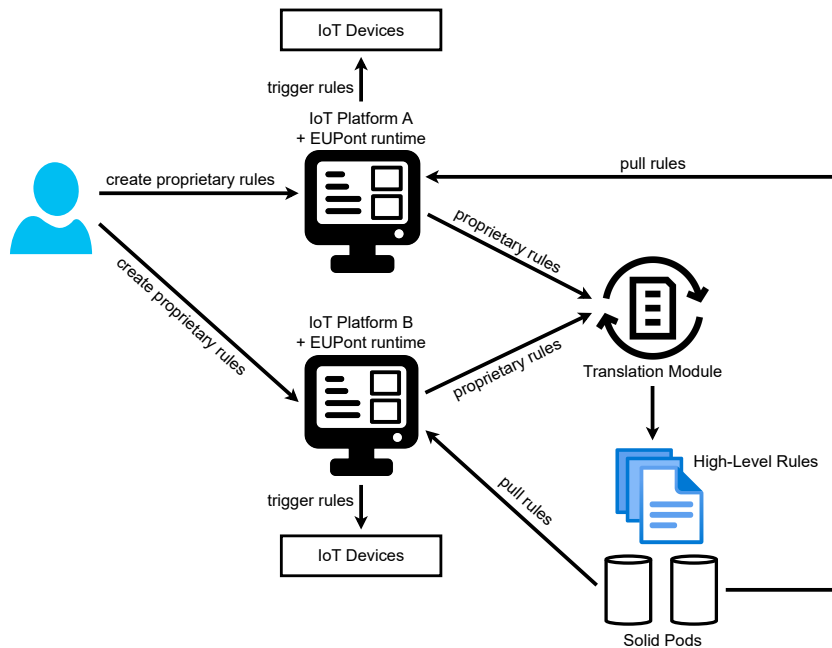


Figure 1: Architecture for translating proprietary IoT rules

access to retrieve it from users’ Pods. The Solid integration thus enables a user to apply their high-level rules on any IoT platform by granting individual platforms access to their Pod. This, in combination with the translation step, leads to the *Run Anywhere* part of our proposed solution. A user can write their rules using a proprietary platform and these rules are then translated by our Translation Module and stored in the user’s Solid Pod. A platform can then pull the user’s translated rules from their Solid Pod and, based on the EUPont runtime, trigger the required actions on the corresponding IoT devices. Further, users are in control of their data, with their Solid Pods as the single access point and source of truth for their IoT rules.

With the proposed translation solution, we aim to minimise or eliminate a user’s need to search for and/or understand the equivalent EUPont representation for their proprietary rules. To achieve this, we first propose that an expert user should be kept in the loop during the testing phase such that initially, they might manually select the best matching translation in situations where the one proposed by the system is inadequate. With this method, we intend that the best results are learned over time and proposed to the user. Due to the popularity of the IFTTT platform, we have selected IFTTT rules as the first type of input to be addressed by our translation method. However, in the future, we intend to apply our approach to other IoT platforms such as Home Assistant<sup>3</sup>.

<sup>3</sup><https://www.home-assistant.io>

As mentioned earlier, the dataset of (Mi et al., 2017) contains 279 828 user recipes, implying that each of those recipes would need to be created for each new platform. With our proposed solution, these recipes can not only be written just once and then be automatically translated to run anywhere, but a recipe’s author can use their preferred authoring tool and keep the ownership of their data (rules). In the remainder of this paper, we describe the translation approach depicted as Translation Module in Figure 1.

### 3.1 Dataset

(Mi et al., 2017) collected published IFTTT recipes (rules) from November 2016 until May 2017. For our automatic translation of rules, we decided to use the most recent May 2017 dataset containing a total of 279 828 recipes (rules). Not each of the recipes necessarily contains unique triggers and actions; the trigger “*Any new photo by you*” is for instance used 9680 times in the dataset. Therefore, the 279 828 rules of the dataset consist of a total of 1017 different triggers and 616 different actions of which 154 triggers and 114 actions appear only *once*. For our proof-of-concept implementation and evaluation, we ran our solution on all different triggers and actions.

### 3.2 Data Preparation

Before applying our translation technique, we performed some data cleaning in order to remove any

present anomalies and to prepare it to be used for the translation steps. For the recipes (rules) present in the dataset (Mi et al., 2017), we noticed that some of the triggers and actions contained a forward slash character. We thus removed that character from the triggers and actions and further separated the triggers and actions into two different lists which we refer to as the *IFTTT dataset*. We also transformed the ontology proposed by (Corno et al., 2019) from XML to JSON to extract its high-level triggers and actions. We removed the redundant *Trigger* and *Action* suffixes from the triggers and actions and separated the high-level trigger and action names into two different lists which we refer to as the *EUPont dataset*.

### 3.3 Translation Technique

Our aim for the translation was to take a rule written by a user in a proprietary format (e.g. IFTTT) and return a generalisation of that rule in the high-level EUPont format by (Corno et al., 2019) that is as accurate as possible. This would enable users to maintain their use of the IFTTT platform without having to learn a new rule description language. To perform this automatic translation from proprietary rules to high-level rules, we apply some natural language processing (NLP) techniques. According to (Qurashi et al., 2020), measuring text similarity is an important part of NLP applications, such as information retrieval, machine translation and text summarisation. For our translation, we applied different document similarity algorithms to both, the IFTTT and EUPont datasets, using the algorithm shown in Listing 1.

```

for each trigger x in EUPont dataset:
  for each trigger y in IFTTT dataset:
    run document_similarity(x,y)
return x, y, Similarity(x,y)
order by similarity descending

for each action a in EUPont dataset:
  for each action b in IFTTT dataset:
    run document_similarity(a,b)
return a, b, Similarity(a,b)
order by similarity descending

```

Listing 1: Pseudocode of translation algorithm

We used the algorithm with three implementations (spaCy, AllenNLP and combined similarity) of the document\_similarity(x,y) function shown in Listing 1 and compared the results as described.

**spaCy Similarity:** The free spaCy<sup>4</sup> open source Python library for advanced Natural Language Processing can be used to build information extraction,

natural language understanding systems or even to pre-process text for deep learning. We use spaCy’s *similarity* feature to compare how similar a given IFTTT trigger and action are to the high-level triggers and actions in the EUPont ontology. We then return the IFTTT trigger or action name together with the computed similar EUPont trigger or action names, as well as the corresponding similarity level.

An example of a result is shown in Listing 2, where the first entry has the EUPont trigger “*Every Time*” returned by the spaCy approach for the IFTTT trigger name “*Any event starts*”. Since not all translations returned by the algorithm are relevant, we defined a *threshold value* which is used to filter out results whose similarity falls below that value. While the threshold is customisable, based on our initial analysis we set its value to 0.55.

```

[
  {
    "Every Time": {
      "ifttt_name": "Any event starts",
      "similarity": 0.7474772725000891 }
  },
  {
    "Every Day": {
      "ifttt_name": "Any event starts",
      "similarity": 0.7034427432691928 }
  },
  ...
]

```

Listing 2: spaCy IFTTT trigger translation example

**AllenNLP Similarity:** AllenNLP (Gardner et al., 2017) is an entire platform for solving NLP tasks and comes with a Python library. We applied the *textual entailment* feature of AllenNLP which, for a pair of sentences, predicts whether the facts in the first sentence imply the facts in the second. We thus determine the textual entailment between each IFTTT trigger and action in the dataset, and the high-level EUPont (Corno et al., 2019) triggers and actions. The AllenNLP textual entailment algorithm returns *entailment* (a measure of the similarity of both texts), *contradiction* (a measure of the dissimilarity of both texts) and *neutral* (a measure of the neutrality of both texts). We return the IFTTT triggers and actions together with the computed similar EUPont triggers and actions as well as their entailment, contradiction and neutral values as illustrated in Listing 3.

**Combined Similarity:** Our initial analysis revealed that the spaCy approach returns more reliable results than the AllenNLP approach. In order to improve the translation results and reduce any noise, we defined a new approach where the AllenNLP algorithm is used to compare the similarity between the initial spaCy re-

<sup>4</sup><https://spacy.io/usage/spacy-101>

sults and the EUPont triggers and actions. For example, for the IFTTT trigger “Any event starts”, we see that the first spaCy result returned is “Every Time” while the first AllenNLP result returned is “Taken”.

```
[
  [
    {
      "ifttt_name": "Any event starts",
      "eupont_hypothesis": "Taken",
      "allen_nlp_entailment": 92.16328263282776,
      "allen_nlp_contradiction":
      3.0818356201052666,
      "allen_nlp_neutral": 4.75488156080246
    },
    {
      "ifttt_name": "Any event starts",
      "eupont_hypothesis": "Received",
      "allen_nlp_entailment": 91.5201187133789,
      "allen_nlp_contradiction":
      3.172384202480316,
      "allen_nlp_neutral": 5.307500064373016
    },
    ...
  ]
]
```

Listing 3: AllenNLP IFTTT trigger translation example

While the first result returned by the spaCy approach might be acceptable, the result returned by the AllenNLP approach is not. However, while conducting our preliminary analysis, we noted that there were more accurate matches further down in the list of results returned by the spaCy approach. We therefore ran the AllenNLP algorithm using the initial spaCy results and the EUPont triggers in order to further improve the translation.

In Listing 4 we can see that the first result returned using this combined approach is “Started Activity”. This result is obtained by combining (averaging) its original spaCy similarity value (61.39) with the similarity value obtained when using the AllenNLP algorithm (85.80). The entry therefore has a combined similarity of 73.60. We are thus able to move this specific spaCy result—which had initially a low ranking compared to the initial “Every Time” top result—to the top. We perform this process for each result returned by the spaCy approach and therefore the results with a high combined similarity are most likely to be the most accurate since they have both, high spaCy and high AllenNLP similarities. An example of the result obtained using this combined approach is highlighted in Listing 4.

While there are several NLP libraries for the Python programming language, such as scikit-learn<sup>5</sup>

<sup>5</sup>[https://scikit-learn.org/stable/tutorial/text\\_analytics/working\\_with\\_text\\_data.html](https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html)

and PyTorch<sup>6</sup>, we opted for the spaCy<sup>7</sup> and AllenNLP libraries, mainly due to their user friendliness. spaCy is also a popular choice in NLP tasks given its fast execution time and the ease with which it lets users build solutions. Similarly, AllenNLP enjoys popularity due to its fast execution time and ease with which it lets a user build prototypes.

```
[
  {
    "ifttt_name": "Any event starts",
    "eupont_hypothesis": "Started Activity",
    "spacy_similarity": 61.39593233371522,
    "allen_nlp_entailment": 85.80678701400757,
    "allen_nlp_contradiction": 3.3948026597499847,
    "allen_nlp_neutral": 10.798408836126328,
    "combined_similarity": 73.6013596738614
  },
  {
    "ifttt_name": "Any event starts",
    "eupont_hypothesis": "Position Registration",
    "spacy_similarity": 57.52355435396419,
    "allen_nlp_entailment": 81.54605627059937,
    "allen_nlp_contradiction": 5.951366946101189,
    "allen_nlp_neutral": 12.50256896018982,
    "combined_similarity": 69.53480531228178
  },
  ...
]
```

Listing 4: Combined IFTTT trigger translation example

## 4 RESULTS

We recorded and compared the results we obtained when applying each of the three different approaches described in the previous section on recipes of the Mi et al. (Mi et al., 2017) dataset. With our translation technique, we intended that the first result recommended by each approach should produce the most accurate high-level EUPont generalisation of the IFTTT triggers and actions. However, in situations where this is not the case, a user should at least be able to find an accurate high-level EUPont generalisation within the first five returned results. We decided to consider only the first five results by each approach to reduce the potential burden a user might face when looking for the best result. In our results, we mark an entry with “No result” if no suitable match has been returned as part of the first five results. For trigger or action names that we consider to be ambiguous—any trigger or action whose meaning could have multiple interpretations—we mark the entry and the resulting translations as “Ambiguous”. For instance,

<sup>6</sup><https://pytorch/nlp.readthedocs.io/en/latest/>

<sup>7</sup><https://spacy.io>

No.	IFTTT Name	spaCy 1	spaCY 2	AllenNLP 1	AllenNLP 2	Combined 1	Combined 2
1	...	...	...	...	...	...	...
2	AC turned off	Device Turned Off	Device Turned Off (1)	Brightness Decreased	No result	Device Turned Off	Device Turned Off (1)
23	Action Button Pressed	Tap Button Activity	Tap Button Activity (1)	Taken	No result	Tap Button Activity	Tap Button Activity (1)
...	...	...	...	...	...	...	...

Table 1: IFTTT trigger translation results

No.	IFTTT Name	spaCy 1	spaCY 2	AllenNLP 1	AllenNLP 2	Combined 1	Combined 2
1	...	...	...	...	...	...	...
10	Add a file	Share File	Save File (2)	Information	No result	Share File	Save File (2)
11	Add a new site (ambiguous)	Connect To Web Service	Ambiguous	Start Focusing	Ambiguous	Save Media Information	Ambiguous
...	...	...	...	...	...	...	...

Table 2: IFTTT action translation results

the trigger “*Air quality changed*” is marked as “*Ambiguous*” because the change in air quality could either be positive or negative. Therefore one approach might return “*Air quality decreased*” as its best result, while another approach might return “*Air quality increased*” as its best result. We further note that several possible acceptable results were returned for certain IFTTT triggers and actions by our approach.

We split our findings into two different tables, with Table 1 showing the results for triggers and Table 2 highlighting the results for actions. Each table contains the following columns:

- **No.:** Entry Number
- **IFTTT Name:** IFTTT trigger or action name
- **spaCy 1:** EUPont trigger or action name with the highest similarity value using the spaCy similarity algorithm.
- **spaCY 2:** EUPont trigger or action name most accurately representing the IFTTT trigger or action name using the spaCy similarity algorithm (position in the result list in brackets).
- **AllenNLP 1:** EUPont trigger or action name with the highest similarity value using the AllenNLP text entailment algorithm.
- **AllenNLP 2:** EUPont trigger or action name most accurately representing the IFTTT trigger or action name using the AllenNLP text entailment algorithm (position in the result list in brackets).
- **Combined 1:** EUPont trigger or action name with the highest similarity value using the combined approach.
- **Combined 2:** EUPont trigger or action name most accurately representing the IFTTT trigger or action name using the combined approach (position in the result list in brackets).

For the presented results, we randomly selected 50 triggers and actions from the results we obtained from running our approaches on the dataset. The results from these 50 triggers and actions were then manually analysed (e.g. to identify and label the most accurate EUPont triggers and actions) in order to populate the entries in Table 1 and Table 2. Note that only a few representative entries from these two tables are shown but the entire tables as well as the complete but non-annotated results of the presented approaches are available online (Attoh and Signer, 2023b).

## 4.1 Analysis

We sought to determine which of the three approaches is performing best. Thereby, we consider an approach to be performing better than another approach based on a combination of the following criteria:

- It has more top results than the other approach.
- It has more top 5 results than the other approach.
- It has fewer cases where a translation could not be found in the top 5 results than the other approach.

In our analysis, entries marked as *ambiguous* (12 out of the 50 randomly selected triggers) were not considered. Therefore, we found that for the remaining 38 triggers, our combined approach returned the best EUPont match as the first result for 16 of those triggers as summarised in Table 3. For 13 of the triggers, the best EUPont match was not the first result but could be found in the top 5 results. However, for 9 of the IFTTT triggers, a suitable EUPont match could not be found by our combined approach. Using the spaCy approach, we found that the best EUPont match was returned as the first result for 13 triggers, while for 9 triggers, the best EUPont match was not the first result but could be found in the top 5 results. However, for 16 triggers, a suitable EUPont match

Approach	First Result	Top Five Result	No result
spaCy	13	9	16
AllenNLP	2	12	24
Combined	16	13	9

Table 3: Summary of trigger translation results

Approach	First Result	Top Five Result	No result
spaCy	10	8	22
AllenNLP	1	11	28
Combined	8	19	13

Table 4: Summary of action translation results

could not be found. Similarly, using the AllenNLP approach we see that the best EUPont match was returned as the first result for 2 triggers. For 12 of the triggers, the best EUPont match was not the first result but could be found in the top 5 results, while for 24 of the triggers no suitable EUPont match was found.

For the actions, there were 10 entries marked as *ambiguous*. Table 4 shows that when using our combined approach on the 40 considered actions, for 8 of the actions the best EUPont match was returned as the first result, while for 19 of the actions the best EUPont match was not the first result but could be found in the top 5 results and for 13 of the actions, no suitable EUPont match could be found. Using the spaCy approach, 10 of the actions had the best EUPont match returned as the first result, while for 8 of the actions the best EUPont match was not the first result but could be found in the top 5 results. However, for 22 of the actions, a suitable EUPont match could not be found. Using the AllenNLP approach, we see that only for 1 out of the actions the best EUPont match was returned as the first result, while for 11 of the actions the best EUPont match was not the first result but could be found in the top 5 results and for 28 of the actions, no suitable EUPont match could be found.

Based on these results, we can conclude that our combined approach is the best-performing approach using our test dataset for both triggers and actions. For triggers, our combined approach returns the highest number of top results, top 5 results and has the smallest number of cases where no result has been returned as part of the top 5. For the actions though, the spaCy approach returns more top results than our combined approach. However, spaCy returns significantly fewer top 5 results and has a larger number of cases where no result was returned as part of the top 5.

## 5 User Evaluation

(Corno et al., 2019) conducted a user study to evaluate the suitability and understandability of the EUPont

approach by end users. The study was a controlled in-lab experiment that involved 30 participants, with only 15 of them having programming experience. It focused on the creation of IoT applications both with the current low-level representation of IFTTT and the high-level representation of EUPont. The study addressed the research questions “*Does the EUPont representation help users create their IoT applications more effectively and efficiently compared to the low-level representation?*” and “*Which of the two representations is preferred by users, and which are the perceived advantages and disadvantages of the two solutions?*”. In summary, the results demonstrated that the EUPont representation allowed end users to reduce the errors and time needed to compose their IoT applications, and introduced numerous benefits in terms of understandability and ease of use.

Based on the findings of the user study by (Corno et al., 2019), we consider the EUPont representation to be a suitable high-level representation of IoT rules for end users. To further evaluate our approach and gather some feedback for future work, we conducted a survey targeting several respondents who were already familiar with the use of IoT automation solutions. We aimed to investigate whether real IoT users would find the results returned by any of our methods to be good high-level generalisations of the IFTTT triggers and actions included in the dataset described earlier in Section 3. The research question we sought to answer with this survey was “*Are the EUPont translations returned by our methods acceptable to end users?*” We had a total of 6 respondents (three male and three female), aged between 20 and 39 years with one of them having obtained a Bachelor’s degree and the others a Master’s degree. In our survey, the respondents were presented a series of IFTTT triggers and actions with their corresponding translations based on the three approaches described in Section 3. They then had to select which method they thought returned a good high-level generalisation of the IFTTT trigger or action and also specify to which degree they found that generalisation to be



accurate. They could further select the “N/A” option in case they found that none of the methods returned a suitable generalisation. For example, given the IFTTT trigger “*New photo upload on page*”, users were presented with the following four options:

- Method 1: “Shared Profile Update”
- Method 2: “No Result”
- Method 3: “Shared Post”
- “N/A”

In a follow-up question, they were asked “*To which degree is your chosen method an accurate generalisation of the IFTTT trigger?*” and had the choice of “*Not at all accurate*”, “*Low accuracy*”, “*Accurate*” and “*Very accurate*”.

For situations where multiple methods returned the same value, the respondents were asked to pick any of those methods if they considered the value to be an accurate generalisation of the trigger or action. For example, for the IFTTT trigger “*If new post from search...*”, method 1 and method 3 returned “*If shared post...*” and respondents could choose any of those two methods if they found it a good high-level generalisation of the IFTTT trigger. We followed the same principle as described in Section 3.3 by considering only the first five results of each method.

The triggers and actions selected for the survey were those we consider to be popular in the dataset based on the fact that they were used 1000 or more times in the dataset described in Section 3. Triggers and actions for which the three methods returned no suitable or ambiguous results were not selected. The survey thus comprised questions for 31 triggers and 33 actions. There were 19 out of the 31 triggers and 19 out of the 33 actions where more than one method returned the same result. We will refer to these cases as *triggers with the same result* and *actions with the same result* respectively. For 12 triggers and 14 actions none of the three methods returned the same result. We will refer to these cases as *triggers with different results* and *actions with different results*.

For the 19 triggers with the same result, in 16 cases at least half of the respondents selected that result as a good high-level description, while for the other 3 cases they indicated that there was no suitable generalisation. In the case of the 19 actions with the same result, at least half of the respondents selected that result as a good high-level description in 18 cases, while for the other single case they indicated that there was no suitable generalisation. For the 12 triggers with different results, there were six triggers where our combined method’s result was not selected. For those six triggers, the result returned by the spaCy method was selected by all respondents;

however, there were five cases where none of the other two methods returned a result for those six triggers and one case where our combined method returned a result which was selected by two respondents. There were 6 of the 12 triggers with different results where the result of the combined method was selected by the respondents. At least half of the respondents selected the result returned by the combined method in 4 out of those 6 cases, while for the other 2 cases, only one respondent selected the result returned by the combined method. For 5 out of the 6 triggers where the result of our combined method was selected by the respondents, the other methods did not return any suitable result while for 1 of the triggers the spaCy method returned a result which was selected by one respondent.

In the case of the 14 actions with different results, there were 8 actions where our combined method’s result was not selected. For those 8 actions, the result returned by the spaCy method was selected in 7 cases by at least half of the respondents and for the last case, the result returned by the AllenNLP method was selected once by at least half of the respondents. Our combined method returned a result for 4 of those eight actions and it was selected by less than half of the respondents in only two cases while AllenNLP returned a result for only one of the 8 actions which was selected by more than half of the respondents. There were 5 of the 14 actions with different results where the result returned by the combined method was selected by the respondents. In 4 cases, less than half of the respondents selected this result, while for the last case half of the respondents selected the result. In 4 of the cases, the other two methods did not return a suitable result, while for the last case, the spaCy method returned a result which was not selected by any respondent. In summary, we can conclude that for 25 out of the 31 triggers and 24 out of the 33 actions, a majority of the respondents selected the result returned by one of our methods as a good high-level generalisation. For the remaining cases, no respondents selected the result returned by one of our methods as a good high-level generalisation.

## 6 CONCLUSIONS

We presented a *Write Once Run Anywhere* paradigm for end-user authoring in IoT settings, helping users to maintain their preferred authoring tool as well as their preferred description language when defining IoT rules that will work across different IoT platforms. To achieve this, we employed the use of natural language processing techniques to automatically translate proprietary rules to high-level EUPont rules

which have been positively received as shown by (Corno et al., 2019). Therefore, we used two popular NLP algorithms in two different approaches and proposed a third novel approach by combining these two algorithms, and carefully analysed the results obtained from all three approaches. The results of our analysis show that all three methods return good high-level generalisations with our novel combined method performing better than the other two methods for the given dataset. We acknowledge that only a small number of users completed our survey, but from these preliminary results, we see that real IoT users also identified the results returned by our three approaches as good high-level generalisations. In future work, we will investigate how to consistently return the most accurate high-level generalisation for a user’s rules, by either using one or a combination of the presented methods. We also plan to focus on improving the accuracy of the results so that more often the first result returned is the most accurate high-level generalisation and the number of cases where no result is returned is minimised or even completely eliminated. Finally, we are going to investigate how our solution can best be integrated into existing IoT platforms as illustrated in Figure 1 to further evaluate the proposed NLP-based rule translation approach with end users.

## REFERENCES

- Attoh, E. and Signer, B. (2021). A Middleware for Implicit Human-Computer Interaction Across IoT Platforms. In *Adjunct Proc. of UbiComp 2021*.
- Attoh, E. and Signer, B. (2023a). From Proprietary to High-Level Trigger-Action Programming Rules: A Natural Language Processing Approach. Technical Report WISE-2023-01, Vrije Universiteit Brussel.
- Attoh, E. and Signer, B. (2023b). Transforming Proprietary to High-Level Trigger-Action Programming Rules Dataset. <https://doi.org/10.5281/zenodo.10033916>.
- Barricelli, B. R. et al. (2019). End-User Development, End-User Programming and End-User Software Engineering: A Systematic Mapping Study. *Journal of Systems and Software*, 149.
- Cabitza, F. et al. (2017). Rule-based Tools for the Configuration of Ambient Intelligence Systems: A Comparative User Study. *MTAP*, 76(4).
- Corno, F., De Russis, L., and Monge Roffarello, A. (2021). Devices, Information, and People: Abstracting the Internet of Things for End-User Personalization. In *Proc. of IS-EUD 2021*.
- Corno, F., De Russis, L., and Roffarello, A. M. (2019). A High-Level Semantic Approach to End-User Development in the Internet of Things. *International Journal of Human-Computer Studies*, 125.
- Corno, F. et al. (2020a). HeyTAP: Bridging the Gaps Between Users’ Needs and Technology in IF-THEN Rules via Conversation. In *Proc. of AVI 2020*.
- Corno, F. et al. (2020b). TAPrec: Supporting the Composition of Trigger-Action Rules Through Dynamic Recommendations. In *Proc. of IUI 2020*.
- Coutaz, J. and Crowley, J. L. (2016). A First-Person Experience with End-User Development for Smart Homes. *IEEE Pervasive Computing*, 15(2).
- Desolda, G. et al. (2016). End-User Development for the Internet of Things: EFESTO and the 5W Composition Paradigm. In *Proc. of RMC 2016*.
- Gardner, M. et al. (2017). AllenNLP: A Deep Semantic Natural Language Processing Platform. In *Proc. of NLP-OSS*.
- Huang, T.-H. K., Azaria, A., and Bigham, J. P. (2016). InstructableCrowd: Creating IF-THEN Rules via Conversations With the Crowd. In *Proc. of CHI 2016 EA*.
- Jeong, H. et al. (2019). Big Data and Rule-based Recommendation System in Internet of Things. *Cluster Computing*, 22(1).
- Krüger, J. (2018). When to Extract Features: Towards a Recommender System. In *Proc. of ICSE 2018 Companion*.
- Li, T. J.-J. et al. (2017). Programming IoT Devices by Demonstration Using Mobile Apps. In *Proc. of IS-EUD 2017*.
- Longo, C. F. et al. (2022). Towards Ontological Interoperability of Cognitive IoT Agents Based on Natural Language Processing. *Intelligenza Artificiale*, 16(1).
- Markopoulos, P. et al. (2017). End-User Development for the Internet of Things. *TOCHI*, 24(2).
- Mattioli, A. and Paternò, F. (2021). Recommendations for Creating Trigger-Action Rules in a Block-based Environment. *Behaviour & Information Technology*, 40.
- Mi, X. et al. (2017). An Empirical Characterization of IFTTT: Ecosystem, Usage, and Performance. In *Proc. of IMC 2017*.
- Nguyen, A. T. et al. (2016). API Code Recommendation Using Statistical Learning From Fine-Grained Changes. In *Proc. of FSE 2016*.
- Noura, M., Atiquzzaman, M., and Gaedke, M. (2019). Interoperability in Internet of Things: Taxonomies and Open Challenges. *Mobile Networks and Applications*, 24(3).
- Ospan, B. et al. (2018). Context Aware Virtual Assistant With Case-based Conflict Resolution in Multi-User Smart Home Environment. In *Proc. of CoCoNet 2018*.
- Qurashi, A. W., Holmes, V., and Johnson, A. P. (2020). Document Processing: Methods for Semantic Text Similarity Analysis. In *Proc. of INISTA 2020*.
- Sambra, A. V. et al. (2016). Solid: A Platform for Decentralized Social Applications Based on Linked Data. Technical report, MIT CSAIL & Qatar Computing Research Institute.
- Trullemans, S., Van Holsbeeke, L., and Signer, B. (2017). The Context Modelling Toolkit: A Unified Multi-layered Context Modelling Approach. *PACMHCI*, 1(EICS).
- Ur, B. et al. (2016). Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes. In *Proc. of CHI 2016*.