Automated Extraction of IoT Critical Objects from IoT Storylines, Requirements and User Stories via NLP

Cristóvão Iglesias (300172291) Rongchen Guo (300228601)

Abstract—**Design an resilient IoT application is one way to deal with threats. However, identify IoT critical objects (services, device and resource) from storyline, requirements and user stories is not an easy and fast task. It is useful to help address resilience in the early stages of the development, but it is time-intensive. The main objective of the project is to assess the usefulness of Named Entity Recognition (NER) models to identify IoT critical objects as a way to make a modelling process less plausible for error and faster. This was performed with development of 5 different machine learning based NER models that were trained and tested with a large dataset with 7396 annotated sentences. We conclude that all developed NER models can identify IoT critical objects and can be useful to support the time-intensive steps of the early stages of the development of IoT systems.**

Index Terms—**Named Entity Recognition, IoT Storyline, natural language processing**

I. Introduction

IoT applications are highly susceptible to threats and one way to deal with threats is to design an resilient IoT application. But resilience should be addressed in the early stages of the development [13, 6]. It should be done in the design phase and the authors of [28] present a way of doing this with Architecture Design Decisions (ADDs). The work [28] presents the Architectural Design Decision Model for Resilient IOT Applications (ADDM4RIOTA) available in github[1]. It is a meta-model used to guide the design of resilient IoT applications. It provided a systematic definition of resilient IoT application and its requirements. Also, the attacks and defense are also discussed. The work proposes a design model for resilient IoT application with the analogy to the human immune system. The work proposes five essential requirements for resilient applications, which are monitoring, detection, protection, restoration and memorization. Based on that, the authors propose their design paradigm which implements these five properties in their design decisions. The main goal of ADDM4RIOTA is to provide a model to aid in dealing with resilience in the early stages of the development of IOT applications. This is important because it allows: i) to deal with the complexity of the problems, ii) effective communication, iii) complete understanding, iv) reduce costs, v) predict behavior, vi) reuse and vii) analyze the feasibility (financial and practical) of the system. In practice, the modeling process, Figure 1, is performed in four phases: first, the IOT application is modeled, second, threats and critical objects (Services, Resources and Devices) are identified (the definitions of these entities can be seen in Table I), third, possible countermeasures are determined, and fourth, countermeasures are selected and it is checked whether updates to the IOT application must be made.

[1]https://cristovaoiglesias.github.io/add4riot/index.html

Although proposed modelling process in [28] is well defined, it needs to be improved to avoid modeling errors and reduce the time-consume. An approach to improve the proposed modelling process is to perform an automated extraction of IoT critical objects (in the second phase) from IoT Storylines, Requirements and User Stories via a Named entity recognition (NER) model, also called entity identification or entity extraction. NER is a natural language processing (NLP) technique for information extraction and retrieval from unstructured texts such as newspapers, blogs and emails. NER automatically identifies named entities in a text and classifies them into predefined categories [26, 29]. The inclusion of the NER model in the second phase of the modelling process has two significant advantages: First, it avoids the drawbacks caused by the manual extraction of the IoT critical objects from the domain models created in the first phase. The domain models are created from storylines, but this activity suffers from the following drawbacks: wrong use of the concepts due to different interpretations of the same text and time consumer. The use of NER in the modelling process is useful, because it sidesteps the first phase. The IoT critical objects can be identified directly from the Storyline, Requirements and User Stories. These last are gaining momentum with the increasing adoption of agile development practices such as Scrum [20, 12]. Second, it allows automation of other activities in the modelling process helping to reduce the time-consume. After identifying the IoT critical objects with the a NER model, we can automatically identify the possible IoT threats and resilient countermeasures.

These advantages of the use of NER make the modelling process less plausible for error and faster. Consequently, this facilitates the use of ADDM4RIOTA to include resilience in the early stages of the development of IoT applications. Furthermore, the work [28] was not published yet and these improvements will be a significant contribution. Improving the modelling process of the proposed approach in [28] is relevant to software engineering, because this work brings five contributions: First, the requirements for modelling a

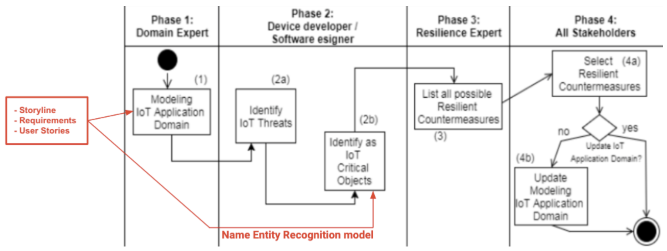


Fig. 1. The UML Activity Diagram illustrating the modelling process of ADDM4RIOTA with its phases and actors (or stakeholders). Phase 1 is done from the use of Storyline, Requirements and User Stories. The main idea of this project is to use NER to sidestep phase 1.

Resilient IoT application are presented. Second, it presents a definition for Resilient IoT Applications based on the resilience requirements raised. Third, a Meta-Model to define a common understanding of a field of interest from the point of view of resilience, through the definition of its vocabulary and key resilient constraints. Fourth, it presents a modelling process to use the ADDM4RIOTA to design a resilient IoT application. This modelling process allows the separation of responsibilities between the different experts involved in the construction of an IoT application. Finally, this approach is a practical way to deal with resilience in the early stages of the development of IoT applications. This last contribution will be possible with the use of automated extraction of IoT critical objects (in the second phase) from IoT Storylines, Requirements and User Stories via a Named entity recognition (NER) model.

This work is aimed at assessing the usefulness of NER model to identify IoT critical objects (services, device and resource) from storyline, requirements and user stories to improve the modeling process of ADDM4RIOTA. This was performed with development of 5 different machine learning based NER models that were trained and tested with a large dataset with 7396 annotated sentences. We conclude that all developed NER models can identify IoT critical objects effectively and can be useful to support the first and second time-intensive phase of the modelling process of ADDM4RIOTA.

The rest of this report is organized as follows: Section II discusses the related work. Section III and IV present the 5 different architectures and their implementation for NER. Section V provides the design and results. Section VI outlines the lessons learned. Section VII concludes the project.

II. Related work

Several authors have applied natural language processing (NLP) techniques in requirements engineering (RE) to address multiple tasks, including model synthesis, classification of requirements into functional/non-functional categories, classification of online product reviews, traceability, ambiguity detection, structure assessment, detection of equivalent requirements, completeness evaluation, and information extraction with NER. Researchers are also investigating how to automatically generate models that capture the key elements of natural language requirements [29, 20]. The authors in [14] proposed a technique for generating conceptual models automatically from user stories. They have implemented a Visual Narrator tool based on a selection of state-of-the-art NL processing heuristics. To calculate the precision and recall, the authors ignore attributes and cardinality, and focus on the details of user stories. Similar work was proposed in [17]. Existing NER methods are typically restricted to some basic sets of entities that are not pertinent to NER in the IoT domain [29, 14, 20, 17]. We define 3 IoT critical objects entities (services, device and resource) which cover the essence of the Storylines, Requirements and Users Stories documents regarding IoT applications. Our work is the first to report on automated extraction of IoT critical objects from IoT storylines, user stories and requirements via NER model.

TABLE I
IoT Critical Objects definition [28].

IoT Critical Object	**Definition**	**Example**
Device	Technical physical component (hardware) to monitor or interact with real-world objects.	Mobile Phone, Embedded system, any sensor, actuator, tag or gateway.
Resource	Computational element that gives access to information about, or actuation capabilities on a real-world object.	Device driver, Programming API, Data repositories, data cache on gateway, data on an RFID tag EPCIS repository, ERP database
Service	Software component enabling interaction with resources through a well-defined interface.	Web service or Local service, such as, Alerting, Monitoring, Connecting.

III. Technical Approach

Named Entity Recognition (NER) [5] is the process of identifying named entities in text. In our scenario, there are three named entities: *Device*, *Resource*, and *Service*. NER is essentially a token classification task where every token is classified into one or more predetermined categories. To this end, we adopted five deep learning based architectures for NER in IoT storylines, user stories and requirements. The architectures are *Spacy*, *BERT*, *transformer*, *LSTM-CRF*, and *ELMo*. The subsequent subsections respectively describe the architecture of the five models developed in this work.

A. Pre-trained SPACY

Spacy [22] is an open source library for natural language processing written in Python and Cython. Fig.2 depicts the workflow of Spacy library with a well-established NER function in NLP pipeline. The Spacy NER system contains a word embedding strategy using sub word features and ”Bloom” embeddings [15], and a deep convolution neural network (CNN) with residual connections. The system is designed to give a good balance of efficiency, accuracy and adaptability. The structure of the NER architecture built-in Spacy is fixed,

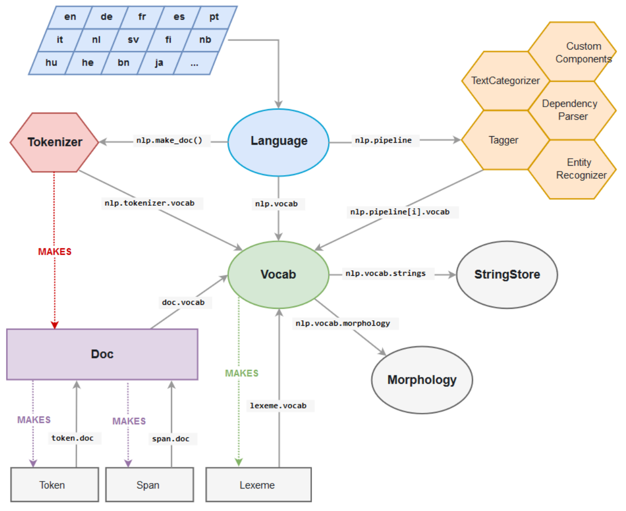


Fig. 2. Spacy library architecture with pretrained NER function.

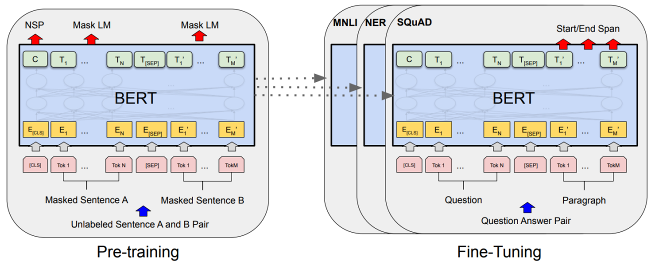


Fig. 3. Overall pre-training and fine-tuning procedures for BERT. Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

but other hyperparameters, such as dropout rate, selection of optimizer and training epochs can be tuned.

B. Pre-trained BERT

Bidirectional Encoder Representations from Transformers (BERT) [21] is the second NER architecture we attempted. BERT is at its core a transformer language model with variable number of encoder layers and self-attention heads.

NER-BERT [27] is the state-of-the-art method for transfer learning [7] in NLP, which consists of two parts: 1) pre-train BERT on a large corpus dataset, and 2) continue training for NER task. Fig.3 demonstrates the transfer learning procedure of BERT. For pre-training, BERT was trained on two tasks: *language modelling* and *next sentence prediction*. As a result of the pre-training process, the model learns contextual embeddings for words. After pretraining, which is computationally expensive, BERT can be fine-tuned with less resources on smaller datasets to optimize its performance on specific tasks. In our case, we fine-tuned BERT on our dataset of 7396 instances for the task of NER.

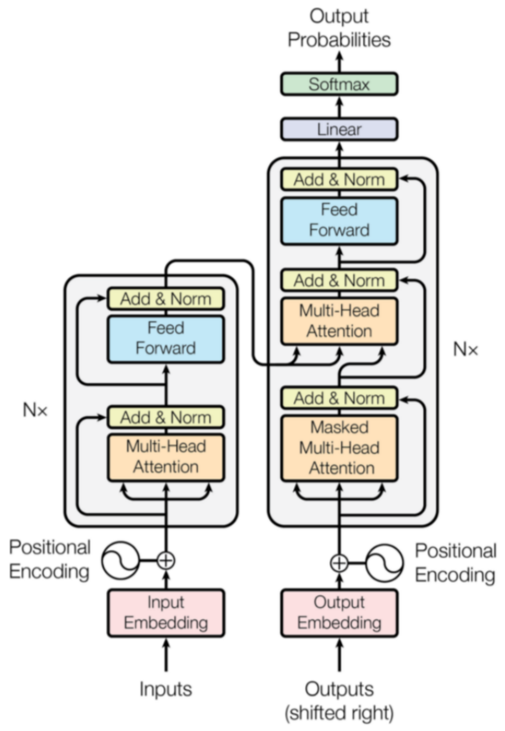


Fig. 4. Model architecture of transformer

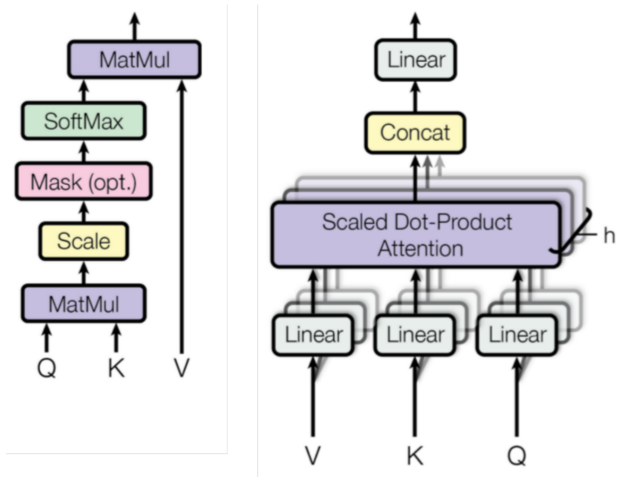


Fig. 5. (a) Scaled Dot-Product Attention. (b) Multi-Head Attention consists of several attention layers running in parallel.

C. Transformers

Transformer [16] is a deep learning architecture that adopts the mechanism of self-attention, differentially weighting the significance of each part of the input data. Fig.4 depicts the architecture of transformers. The model consists of multi-head attention (fig.5) and feed forward layers. The scaled dot-product attention-mechanism (fig.5(a)) is built based on the *seq2seq* autoencoder framework [10], which consists of an encoder and a decoder. The attention mechanism used in the model can be described by the following equation:

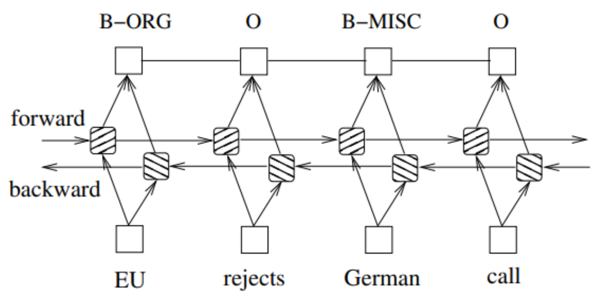


Fig. 6. biLSTM-CRF model structure

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

where Q is a matrix that contains the query (vector representation of one word in the sequence); K are all the keys (vector representations of all the words in the sequence); and V are the values, which are again the vector representations of all the words in the sequence. The values in V are multiplied and summed with attention-weights a, where

$$a = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{2}$$

In other words, the weights a are defined by how each word of the sequence (represented by Q) is influenced by all the other words in the sequence (represented by K). Those weights are then applied to all the words in the sequence that are introduced in V. Additionally, the scaled dot-product attention-mechanism can be parallelized into multiple mechanisms that can be used side by side (fig.5(b)), which largely saves the training time of the model. After the multi-attention heads in both the encoder and decoder, a pointwise feed-forward layer is added. The feed-forward network has identical parameters for each position, which can be described as a separate, identical linear transformation of each element from the given sequence.

D. bidirectional LSTM-CRF

LSTM-CRF [11] is a combination framework of a *Long-Short Term Memory (LSTM)* network [2] and a *Conditional Random Fields (CRF)* network [4]. The model architecture is given in fig.6. This network can efficiently use past input features via an LSTM layer and sentence level tag information via a CRF layer. A CRF layer is represented by lines which connect consecutive output layers. A CRF layer has a state transition matrix as parameters. With such a layer, we can efficiently use past and future tags to predict the current tag, which is similar to the use of past and future input features via a bidirectional LSTM network.

The input sequence to CRF is the tokens of a sentence $x = (x_1, \ldots, x_m)$, and the output states $s = (s_1, \ldots, s_m)$ are the named entity tags that we want to identify. In conditional random fields (CRFs), the conditional probability $p(s_1, \ldots, s_m | x_1, \ldots, x_m)$ of the output state sequence s given

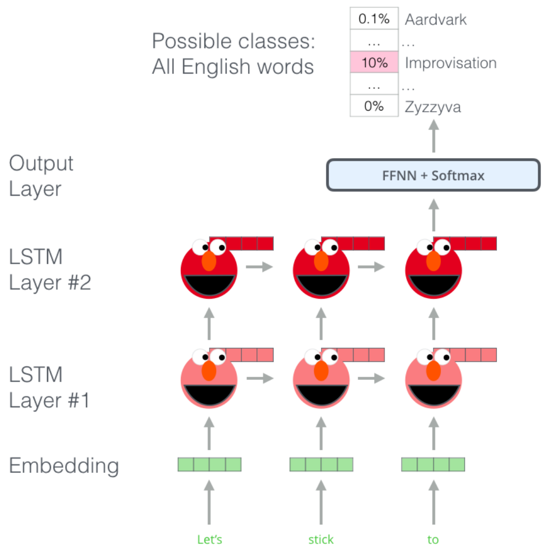


Fig. 7. Model architecture of ELMo

a input sequence$x$ is calculated by defining a feature map $\Phi(x_1, \ldots, x_m, s_1, \ldots, s_m) \in \mathbb{R}^d$ that maps an entire input sequence x paired with an entire state sequence s to some d-dimensional feature vector. Then the conditional probability is modeled as a log-linear model with the parameter vector $w \in \mathbb{R}^d$:

$$p(s|x;w) = \frac{exp(w \cdot \Phi(x,s)}{\Sigma_{s'} exp(w \cdot \Phi(x,s')} \tag{3}$$

where s' ranges over all possible output sequences. The expression $w \cdot \Phi(x,s) = \text{score}_{crf}(x,s)$ is viewed as a scoring how well the state sequence fits the given input sequence. Lastly, the linear scoring function is replaced by LSTM, a non-linear neural network. So the score of LSTM-CRF network can be defined as:

$$score_{lstm-crf}(x,s) = \sum_{i=0}^{n} W_{s_{i-1},s_i} \cdot LSTM(x)_i + b_{s_{i-1},s_i} \tag{4}$$

where W_{s_{i-1},s_i} and b are the weight vector and the bias corresponding to the transition from s_{i-1} to s_i, respectively. Note, that the score functions are also called *potential functions*. After constructing this score function, we can optimize the conditional probability $p(s|x;W,b)$ like in the usual CRF and propagating back through the LSTM network. Moreover, in our implementation, we used bi-directional LSTM instead of a simple LSTM to better more the sequential relationships among tokens in a sentence.

E. ELMo (Embedding from Language Models)

Embeddings from Language Model (ELMo) [19] is a deep contextualized word representation that models both 1) complex characteristics of word use (e.g., syntax and semantics), and 2) how these uses vary across linguistic contexts (i.e., to model polysemy). Unlike traditional word embedding methods, such as *Bag of Words* [3], *Word2Vec* [1] and *GloVe* [9], the word embeddings produced by ELMo are context-sensitive (ie, the embedding of a certain word differs depending on the context). Concretely, the word representations learnt by ELMo has three characteristics: i) Contextual: The representation for each word depends on the entire context in which it is used; ii) Deep: The word representations combine all layers of a deep pre-trained neural network; and iii) Character based: ELMo representations are purely character based, allowing the network to use morphological clues to form robust representations for out-of-vocabulary tokens unseen in training. The word representations are learned functions of the internal states of a deep bidirectional language model (biLM), which is pre-trained on a large text corpus. They can be easily *transferred* to existing models and significantly improve the state of the art across the task on NER. Fig.7 depicts one step in pre-training ELMo.

IV. Implementation

We implemented the five machine learning based NER algorithms described in Sec.III in Python. As shown in fig.10, most sentences in the dataset have less than 50 tokens, so we pad the sentences to the length of 50 to make sure a consistent input size to the models. A masking layer is then added to enable algorithms to distinguish between the padded sentences and original sentences.

As our implementation, *Spacy* is implemented using the NER function built-in the NLP pipeline of spacy library in Python. *BERT* is implemented in PyTorch, *ELMo* in TensorFlow, while *transformers* and *LSTM-CRF* are implemented in keras. All codes of implementations are available in github [2].

V. Empirical Evaluation

The main objective was refined into the 3 research questions.

A. Research Questions

Our evaluation aims to answer the following 3 Research Questions (**RQs**):

- **RQ0**. **How is the performance of a fine-tuned NER model using a small dataset to identify IoT critical objects?** This was the first question to be answered, because a positive result indicates that NER has the potential to identify IoT critical objects, but the confirmation should be found answering next RQs.
- **RQ1**. **Could a large dataset increase the performance of a NER Model?** After answer the RQ0, we need to check if we can improve the performance of model developed (in RQ0) with a large dataset regarding entities service, device and resource.
- **RQ2**. **Which NER architecture has the best performance to identify IoT critical Objects?** After answer the RQ1, we need to compare five different architectures to design NER models. The five architectures are SPACY, BERT, Transformers, Bidirectional LSTM-CRF and ELMo.

[2]https://github.com/cristovaoiglesias/csi5137_course

B. Description of Dataset

It was not possible to find in the literature a public dataset with annotations regarding the entities: services, device and resource to attend to the main objective of this work. Therefore, we developed a large dataset with a total of 7396 annotations, see the Table II. The task to address the main objective requires a dataset with a minimum of 3000 annotations. We are using this number as a reference because a dataset with this size is considered a large-sized dataset. Furthermore, there are published papers using datasets with this size [18, 24] for information extraction tasks. The developed dataset has a total of 2157 annotations for device, 3806 annotations for resource and 1433 annotations for service. This dataset is splitted into training set (with a total of 4807 annotations) and testing set (with a total of 2589 annotations). The testing set was created using 35% of each total of annotations of the IoT critical objects. In addition, among the 7396 annotations, the biggest sentence has 115 tokens, but the majority have around 20 tokens, see Figure 10.

Two approaches were used to develop the large dataset with annotation following the format described in the Figure 8.
Approach 1 - to annotate sentences for the entities of *device* and *resource*, the following processes were adopted:

Process 1 - Source: Online Sentence Dictionaries

1) Identify examples of devices, services and resources from their definitions (Table I).
2) Search sentences using the examples of devices, services and resources in Online Sentence Dictionaries[3],[4].
3) Annotate the sentences with Doccano[5] and export the .json file. Doccano is an open source text annotation tool for humans. It enables the creation of labeled data for named entity recognition in hours.

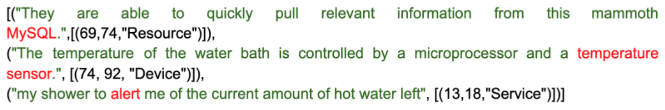
Process 2 - Source: IoT Storylines and Requirements [8]

1) Upload IoT Storylines and Requirements documents in Doccano
2) Annotate the sentences with Doccano and export the .json file.

Approach 2 - to annotate sentences for the entity *service*, CrowdRe[6] Dataset was used. It includes 2966 requirements (sentences) in the form of user stories with role, feature and benefit, well-defined in each sentence, see the Figure 9. Some of the first verbs inside of feature follow the definition of the entity service and could be used as instance of this entity, see Table I and Figure 9. Therefore, the process steps of annotating the CrowdRe Dataset are (using NLTK in python):

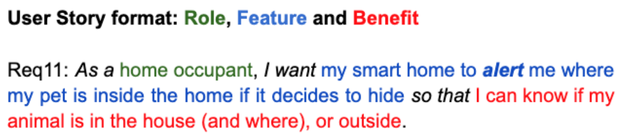
1) Tokenize sentences (strings) in CrowdRe dataset.
2) Perform POS tagging.
3) Got the first verb in the pos_tag array.

[3]https://www.wordhippo.com
[4]https://sentence.yourdictionary.com
[5]https://github.com/doccano/doccano
[6]https://crowdre.github.io/murukannaiah-smarthome-requirements-dataset



[("They are able to quickly pull relevant information from this mammoth MySQL.",[(69,74,"Resource")]),
("The temperature of the water bath is controlled by a microprocessor and a temperature sensor.", [(74, 92, "Device")]),
("my shower to alert me of the current amount of hot water left", [(13,18,"Service")])]

Fig. 8. Example with 3 annotation obtained with the approaches 1 and 2.



User Story format: Role, Feature and Benefit

Req11: *As a* home occupant, *I want* my smart home to ***alert*** me where my pet is inside the home if it decides to hide *so that* I can know if my animal is in the house (and where), or outside.

Fig. 9. CrowdRe Dataset with user story format.

4) Identify the position of this verb with respect to the sentence string.
5) Create the annotation.

Before creating the large dataset with the device, resource, and service, a small dataset with 150 device annotations was developed using approach 1 with process 1. This small dataset was used to answer the **RQ0** and the large dataset was used to answer the **RQ1** and **RQ2**. The large dataset is available on github [7].

TABLE II
DATASET WITH ANNOTATED IoT CRITICAL OBJECTS

		Device	**Resource**	**Service**	**Total**
Dataset Size	**Train**	1402	2474	931	4807
	Test	755	1332	502	2589
	Total	2157	3806	1433	7396

C. Analysis Procedure and Metrics

The analytical procedure to answer each of the research questions in Section V-A will be based on analyzing the precision, recall and F1-score obtained with the NER models generated.

- **Precision**: The proportion of true annotations among all annotations identified.

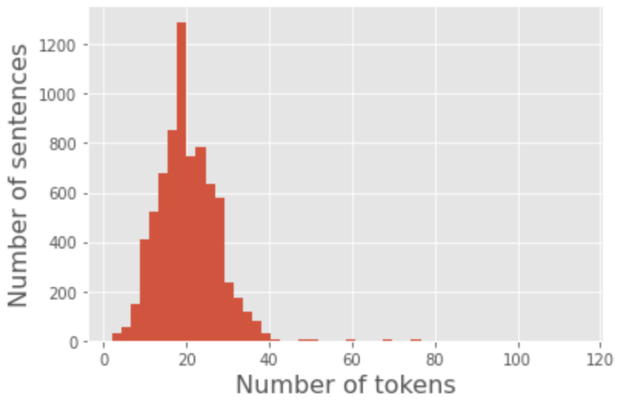


Fig. 10. The biggest sentence has 115 tokens.

[7]https://github.com/cristovaoiglesias/csi5137_course

- **Recall**: The proportion of true annotations identified among all annotations that should have been identified.
- **F1-score**: The harmonic mean of Precision and Recall, an overall metric to evaluate the model.

To answer **RQ1**, we would apply the above metrics to evaluate the performance of NER built-in SpaCy on the large dataset and compare it to that on the small dataset. To answer **RQ2**, we would calculate the precision, recall and f1 score across all models, and compare the value of these metrics on testing set. This horizontal comparison let us understand which model performs the best, and whether there are significant differences among these models. Positive answers for the **RQs** indicates that the NER models are useful to identify IoT critical objects (services, device and resource) from storyline, requirements and user stories and this can be considered an improvement in the modeling process of ADDM4RIOTA [28].

D. Results and Discussion

Below, we present and discuss our results, organized by RQ0, RQ1 and RQ2:

RQ0: We fine-tuned a Spacy NER model with the small dataset with 150 annotations regarding entity device. We hold-out 50 of annotations as a testing set, and use the rest to train the model. Based on precision score of 66.67%, recall of 59.57%, and F1-score of 62.91%, we can say that a fine-tuned NER model learned some information from the small dataset. This was a promising result in the domain of NER, especially when we notice that the recall is quite high, indicating that the fine-tuned NER model is not losing much useful annotations that should have been identified.

RQ1: Here, we fine-tuned a Spacy NER model with the large dataset with 7396 annotations regarding all entities. The performance of this architecture was improved with the training using the large dataset. In Table III, we can see that the model presented precision, recall and f1-score with values higher than model trained with the small dataset (**RQ0**). All values are above 90%. In addition, this model trained with the large dataset also presented a good performance regards resource and service.

TABLE III
MODELS PERFORMANCES WITH LARGE DATASET.

Entity	**Model**	**Precision**	**Recall**	**F1-score**
Device	SPACY	93.9	92.8	93.3
	BERT	93.0	92.0	93.0
	Transformers	92.0	91.0	92.0
	LSTM-CRF	67.0	81.0	74.0
	ELMo	77.0	89.0	83.0
Resource	SPACY	95.8	93.9	94.9
	BERT	90.0	94.0	92.0
	Transformers	96.0	87.0	91.0
	LSTM-CRF	91.0	85.0	88.0
	ELMo	94.0	91.0	92.0
Service	SPACY	87.8	82.5	85.0
	BERT	96.0	88.0	92.0
	Transformers	89.0	67.0	76.0
	LSTM-CRF	88.0	74.0	81.0
	ELMo	95.0	89.0	92.0

RQ2: The five machine learning architectures described in Sec.III are performed on the task of NER with respect to the large dataset described in Sec.V(B). *Precision*, *recall*, and f1 score are used as metrics to evaluate the model performances. Since the dataset contains three entities (*device*, *resource*, and *service*), precision, recall, and f1 score are computed respectively to the three entities. The results are shown in Tab.III.

All five deep learning networks have a good performance identifying the three entities. But, ***BERT* outperforms the other four models in terms of evaluation metrics and also with high stability and robustness.** BERT has above 90% of both precision and recall across all three entities, while the other four models only performs well with respect to certain but not all entities.

Spacy and *transformers* had a good performance identifying *device* and *resource*, but perform less well for *service*. In the large dataset, *service* has a high variance regards the instances of service, while *device* and *resource* are of low variance. This indicates that these two models suffer from identifying entities with a lot of variations. Especially for *transformers*, its recall drops significantly on the class of *service*, as compared to its performance on the other two classes. On the other hand, *LSTM-CRF* performs better in identifying *resource*, and performs less well in identifying *device* and *service*; *ELMo* has an excellent f1-score on *resource* and *service*, but has a low score on *device*.

E. Threats to Validaty

1) Large dataset with sentences and not with text: In general user stories, and non-functional and functional requirements are described in sentences. But the Storylines are described with texts (ie, paragraphs or articles). The models do not have a good performance to identify IoT critical objects from input texts. To side-step this the input text should be processed (splitted) sentences that fit with the input size of the NER models. This is because all NER models were trained with sentences of 50 tokens in maximum.

2) Imbalanced dataset: The dataset developed can be considered large-sized, but it also can be considered imbalanced, since it has 3806 annotations for entity resource and 1433 for entity service. To mitigate this threat to the validity of our empirical analysis we used recall and f1-score beyond accuracy. Accuracy is not a good metric for Imbalanced Classification [23, 25]. Accuracy fails to keep the integrity of the reliability of the outcome for unbalanced data modeling. This problem is known as Accuracy Paradox [23]. The main reason is that the overwhelming number of examples from the majority class (or classes) will overwhelm the number of examples in the minority class, meaning that even unskillful models can achieve accuracy scores of 90 percent, or 99 percent, depending on how severe the class imbalance happens to be [25]. F1-score is an option better than Accuracy for imbalanced classification. F-measure provides a way to combine both precision and recall into a single measure that captures both properties. Alone, neither precision or recall tells the whole story. We can have excellent precision with terrible recall, or alternately, terrible precision with excellent recall. F-measure provides a way to express both concerns with a single score [25]. F1-score might be the most common metric used on imbalanced classification problems [25]. The intuition for F-measure is that both measures are balanced in importance and that only a good precision and good recall together result in a good F-measure [25].

VI. Lessons Learned

The large dataset includes 1 annotation per sentence. For example, the two sentences below have only one annotation:

- *This* ***sensor (DEVICE)*** *is compact and reliable for use in the cover gas hydrogen meter in place of the katharometer.*
- *Doors and windows that* ***notify (SERVICE)*** *me if they are opened while armed.*

The training of the models with a dataset with 1 annotation per sentence does not means that the mode will be able to identify two or more entity instances in the same sentence. For example;

- *As a Doctor, I want to the system to* ***notify (SERVICE)*** *me about abnormality in* ***sensor (DEVICE)*** *readings.*

In the sentence above the models developed probably would identify service or device. Only in few cases the models developed were able to identify two or more entity instances in the same sentence.

VII. Conclusion and Future works

The high values of precision, recall and f1-score indicate a good performance of the NER models to identify IoT critical objects (services, device and resource) from storyline, requirements and user stories and this can be considered an improvement in the modeling process of ADDM4RIOTA. The main contribution of this project were: i) a large sized dataset with annotations regards IoT critical objects and ii) 5 differents NER models to identify IoT critical objects.

More experiments should be performed and the future works can be answering the following research question: **RQ3 - Is it possible to use the identified IoT critical objects to list all possible IoT threats and resilient countermeasures that can be used in the design of an IoT application?** By using the identified IoT critical objects, we can perform an automatic search in the tables proposed in ADDM4RIOTA. In addition, the large dataset can be improved. It can be done in the following way: i) more annotation for service and device, ii) more variability of instances of resource and device, and iii) more annotations per sentence.

References

[1] Marie-José Aldon and Olivier Strauss. “A new shape segmentation approach for active vision systems”. In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society. 1991, pp. 708–709.

[2] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[3] Sam Scott and Stan Matwin. “Text classification using WordNet hypernyms”. In: *Usage of WordNet in Natural Language Processing Systems*. 1998.

[4] John Lafferty, Andrew McCallum, and Fernando CN Pereira. “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”. In: (2001).

[5] David Nadeau and Satoshi Sekine. “A survey of named entity recognition and classification”. In: *Lingvisticae Investigationes* 30.1 (2007), pp. 3–26.

[6] Dong Liu, Ralph Deters, and Wen-Jun Zhang. “Architectural design for resilience”. In: *Enterprise Information Systems* 4.2 (2010), pp. 137–152.

[7] Lisa Torrey and Jude Shavlik. “Transfer learning”. In: *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*. IGI global, 2010, pp. 242–264.

[8] Alessandro Bassi et al. *Enabling things to talk*. Springer Nature, 2013.

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[10] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. *Sequence to Sequence Learning with Neural Networks*. 2014. arXiv: 1409.3215 `[cs.CL]`.

[11] Zhiheng Huang, Wei Xu, and Kai Yu. *Bidirectional LSTM-CRF Models for Sequence Tagging*. 2015. arXiv: 1508.01991 `[cs.CL]`.

[12] Chetan Arora et al. “Extracting domain models from natural-language requirements: approach and industrial evaluation”. In: *Proceedings of the ACM/IEEE 19th International Conference on Model Driven Engineering Languages and Systems*. 2016, pp. 250–260.

[13] Kemal A Delic. “On resilience of IoT systems: The Internet of Things (ubiquity symposium)”. In: *Ubiquity* 2016.February (2016), pp. 1–7.

[14] Marcel Robeer et al. “Automated extraction of conceptual models from user stories via NLP”. In: *2016 IEEE 24th international requirements engineering conference (RE)*. IEEE. 2016, pp. 196–205.

[15] Joan Serrà and Alexandros Karatzoglou. “Getting deep recommenders fit: Bloom embeddings for sparse binary input/output networks”. In: *Proceedings of the Eleventh ACM Conference on Recommender Systems*. 2017, pp. 279–287.

[16] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[17] Meryem Elallaoui, Khalid Nafil, and Raja Touahni. “Automatic transformation of user stories into UML use case diagrams using NLP techniques”. In: *Procedia computer science* 130 (2018), pp. 42–49.

[18] Tim Gemkow et al. “Automatic glossary term extraction from large-scale requirements specifications”. In: *2018 IEEE 26th International Requirements Engineering Conference (RE)*. IEEE. 2018, pp. 412–417.

[19] Matthew E. Peters et al. *Deep contextualized word representations*. 2018. arXiv: 1802.05365 `[cs.CL]`.

[20] M Veera Prathap Reddy et al. “NERSE: Named Entity Recognition in Software Engineering as a Service”. In: *Service Research and Innovation*. Springer, 2018, pp. 65–80.

[21] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 `[cs.CL]`.

[22] Xavier Schmitt et al. “A replicable comparison study of NER software: StanfordNLP, NLTK, OpenNLP, SpaCy, Gate”. In: *2019 Sixth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. IEEE. 2019, pp. 338–343.

[23] Muhammad Fahim Uddin. “Addressing Accuracy Paradox Using Enhanched Weighted Performance Metric in Machine Learning”. In: *2019 Sixth HCT Information Technology Trends (ITT)*. IEEE. 2019, pp. 319–324.

[24] Kushagra Bhatia, Siba Mishra, and Arpit Sharma. “Clustering Glossary Terms Extracted from Large-Sized Software Requirements using FastText”. In: *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly known as India Software Engineering Conference*. 2020, pp. 1–11.

[25] Jason Brownlee. *Imbalanced classification with Python: better metrics, balance skewed classes, cost-sensitive learning*. Machine Learning Mastery, 2020.

[26] Jing Li et al. “A survey on deep learning for named entity recognition”. In: *IEEE Transactions on Knowledge and Data Engineering* (2020).

[27] Zihan Liu et al. *NER-BERT: A Pre-trained Model for Low-Resource Entity Tagging*. 2021. arXiv: 2112.00405 `[cs.CL]`.

[28] Cristovão Iglesias, Claudio Micelli, and Miodrag Bolic. “An Architectural Design Decision Model for Resilient IoT Application (paper not published)”. In: (2022).

[29] Garima Malik et al. “Named Entity Recognition on Software Requirements Specification Documents”. In: ().