Unsupervised Named Entity Recognition for Hi-Tech domain

Abinaya Govindan, Gyan Ranjan, and Amit Verma

Neuron7.ai, USA

Abstract. This paper presents named entity recognition as a multi-answer QA task combined with contextual natural-language-inference based noise reduction. This method allows us to use pre-trained models that have been trained for certain downstream tasks to generate unsupervised data, reducing the need for manual annotation to create named entity tags with tokens. For each entity, we provide a unique context, such as entity types, definitions, questions and a few empirical rules along with the target text to train a named entity model for the domain of our interest. This formulation (a) allows the system to jointly learn NER-specific features from the datasets provided, and (b) can extract multiple NER-specific features, thereby boosting the performance of existing NER models (c) provides business-contextualized definitions to reduce ambiguity among similar entities. We conducted numerous tests to determine the quality of the created data, and we find that this method of data generation allows us to obtain clean, noise-free data with minimal effort and time. This approach has been demonstrated to be successful in extracting named entities, which are then used in subsequent components.

Keywords: natural language processing,named entity recognition unstructured data generation, question answering, information retrieval

1 Introduction

The increasing availability of open source Natural Language Processing (NLP) resources and toolkits, combined with the massive amount of data generated every day, necessitates the development of tools that can analyse this data and extract useful information. Unfortunately, just because there is more data being generated every day does not indicate that it can be used to train modern deep learning systems.

In NLP, named entity recognition (NER) is a crucial task which aims to recognise and classify named items such as persons, locations, and events. These extracted named entities are used in a variety of NLP operations to help make better sense of unstructured data.

Some of the early applications of NER included human name identification in a given system such as [1], question answering models [2] that use entity recognition to improve search results and document summarization systems such as [3], where NER identifies significant parts of text that contribute to summaries.

Neural network-based models have recently improved the performance of NER tasks, due to the advances in deep learning. For various human languages, including English, French, and Chinese, named entity models have been developed.

Since NER is becoming more important across many fields and businesses, domain-specific NER technologies have become the new focus. Several NER models for the medical domain system such as [4], have been created to identify a variety of medical categories, including Genes, chemicals, diseases, and so on. This is due to (a) the availability of open source data for all of these areas, such as [5], (b) These datasets do not have any confidentiality associated and thus suitable for training.

However in some domains, using these architectures may be insufficient because the performance of these models is dependent on the quantity and quality of labeled data, and annotated data generation might be particularly difficult because these models require a large amount of high-quality data. This drives researchers to hope to develop a mechanism for extracting semantic and lexical knowledge from enormous amounts of unstructured, unlabeled data, which can then be applied to the NER task thereby improving the performance.

[6] are creating a Service Intelligence platform that, given a faulty hi-tech hardware, recommends actions and provides actionable insights to the repair technician. While there are commercial applications to create, edit, and search technician notes, historical technician repair notes have not been leveraged to derive insights. A key characteristic of insight recommendation engine is to understand the context of these notes, which is provided by the named entities, and given the unstructured nature of these notes, is not trivial to extract. These extracted named entities a) directly assist technicians by giving focused and most informative segments of the notes, allowing them to spend less time reading and perceiving the notes. b) provide an overview of the problem along with recommendations for parts or locations that the technician should investigate further.

So after careful evaluation based on the above criteria, classes such as *Model name, Parts replaced, Error codes, Frequency, Amplitude, Functional Test performed* are the entity tags that we chose to extract and train.

The goal of this research is to present a system for generating labeled data that can be used directly to train a domain-specific NER model. We perform our experiments on technical case descriptions and technician notes that have been raised on the service intelligence platform. These notes along with the extracted entities provide a technical insight onto what could be the reason behind the exhibited symptom and subsequently the proposed resolution.

From the generated data, we only retain the best quality data, rejecting the ambiguous data points and reducing the noise by employing an ensemble of numerous definitions and business contextualised rules. We also go over the results of fine-tuned models and architectures trained on the generated data.

In this paper, we study various approaches of data generation to train custom named entity modlels. We show our proposed system and novel modules implemented to achieve unsupervised data generation. We also compare the performance of various deep learning models trained on the generated data along with associated results which depict the effectiveness of our system.

1.1 Related work

Due to the domain specific terminologies and ambiguity of these business terms, Named Entity Recognition (NER) has been deemed a relatively challenging problem in the domain of Hi-Tech. Experiments conducted by [7] show that external knowledge can be helpful in classifying the same terms as different named entities depending on the context. Earlier works like [8] have used rule based and dictionary based approaches to solve the NER task for particular domains and languages. However, these approaches have weak generalisation properties when applied to unseen data.

Other learning based systems developed by [9] abd [10] have wide applications across a variety of domains but have some limitations, such as the lack of specific domain knowledge integration and the inability to handle novel entity types with limited data availability. They are also occasionally under-optimized for accuracy due to the usage of less powerful models, resulting in poor performance in downstream activities. For the task of NER, researchers such as [11] , [12] have developed machine learning models such as Hidden Markov Models (HMM) and conditional random fields (CRF). These machine learning methods, however, demand comprehensive and time-consuming feature identification and extraction, which can be costly in terms of manual labour.

Modern deep learning architectures for NER, such as [13], overcome the issues faced by these models. These models generally function best on massive amounts of labeled data, and are thus frequently created for open-source or academic sources. [14] attempts to solve the paucity of data in a few areas by transferring an ANN model trained on a large labeled dataset to a smaller unlabeled dataset. Other synthetic data generation-based methods include (a) back-translation, noise reduction, and parallel sentence extraction as suggested by [15] (b) data labelling modules that use open source websites such as Wikipedia and Google to label data automatically as proposed by [16].

2 Our work

2.1 Task formulation

NER is defined in traditional systems as a token level multi-class classification task. For the purposes of this study, we will concentrate on data generation and transformation into a format that comprises of text tokens and corresponding named entity tags. The named entity tag is created as

$$B - e_k,\, I - e_k \, and \, O$$

where e_k is the entity type. As a result, given any token and context $C_1, C_2, C_3..C_n$, the unsupervised data generation module generates tags $L_1, L_2, L_3...L_n$ which are used to fine-tune a named entity classifier. The tags must be grouped into B and I for us to identify the beginning and end of each entity. We employ four forms of knowledge context based on our experiments: entity types, questions that represent each entity type, definitions for each entity type, and business rules created for each entity type. Due to the lack of innate business domain features and standards, text sequence by itself may not enable us obtain the optimum labels for the provided data, therefore these knowledge contexts are essential to enhance the quality of the data generated.

2.2 Dataset

The data we used for this study was our service intelligence platform's unstructured agent notes and issue descriptions created by technicians. We must be able to give high-quality named entities with a small margin of error because the extracted entities directly assist the technicians in understanding the problem and prescribing solutions. However, due to many limitations in these unstructured notes, such as domain specialised language, a lack of correct linguistic structure, and shorter chunks of text with condensed information, this is not a simple process.

Due to these restrictions, annotated named entities are required to capture all of these information in the tagged data so that the NER model can learn them. To produce named entities from technician notes for diverse product lines, we use various unsupervised approaches such as question answering, natural language inference, and conditional text generation. Not all entity tags have a similar distribution of tagged data, resulting in an imbalance in the data. We accommodate for this imbalance by giving tailored context along with the text sequence to assist the model to learn these features better.

2.3 Candidate generation

The overall approach for data generation is depicted in 1

The first stage of data generation entails extracting candidate named entities using an ensemble of multiple variants of QA models fine tuned on SQuAD [17].We employ an ensemble because we want to minimise the error that a single model can produce, and extract candidates with high confidence using majority voting approaches. This would allow us to reduce the noise produced by these separate models while also maximising their aggregate performance.For each of the mentioned entities, we create a few templates such as:

– *What were the parts replaced* for **Replaced parts** ,
– *What was the error code mentioned in the note* for **Error code**

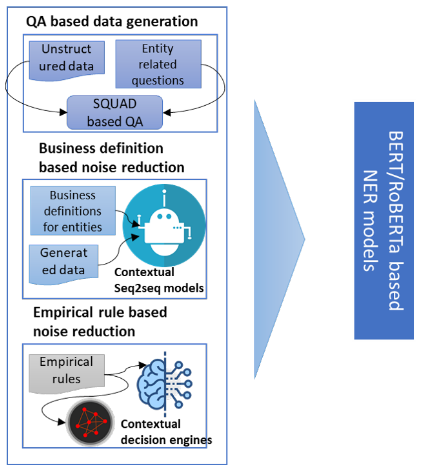
QA based data generation
Unstructured data
Entity related questions
SQUAD based QA
Business definition based noise reduction
Business definitions for entities
Generated data
Contextual Seq2seq models
Empirical rule based noise reduction
Empirical rules
Contextual decision engines
BERT/RoBERTa based NER models

Fig. 1. Architecture for data generation and NER training

- *What was the value of frequency mentioned* for **Frequency**
- *What was the value of amplitude mentioned* for **Amplitude**

During our investigation of the coverage of annotated labels, we determined that a one-to-one question-to-entity type mapping effectively covered these entity tags. For the entity type **Functional test**, we ask a few different variations of questions, such as *What test was performed* , *What was the name of the test that failed etc.* . These question variants enabled us capture a large number of annotated sample for this entity type. In this dataset, these were the best-performing questions during inference.

As demonstrated in [18], we used several models fine tuned on SQuAD2.0 dataset. Using these models in an ensembled fashion helps in more accurate named entities when compared to using a single model, which in turn improves the final NER model's performance.

2.4 Context and definition based candidate noise reduction

Once we've arrived at a list of candidate entities, we finalised a set of carefully crafted definitions for each of the entity tag, based on business expertise that was provided by the technicians and experts. The definitions were

- Frequency is the rate at which current changes direction per second. Frequency is measured in Hertz (Hz)
- Amplitude is the maximum displacement or distance moved by a point on a vibrating body or wave measured from its equilibrium position. I. Amplitude is measured in decibels (dBs)

– A functional test refers to an operational test performed on a machine that indicates which part of a machine fails.

We incorporate these definitions for additional noise reduction by providing this as a context to a text generation model which was fine tuned to answer Boolean questions as in [19]. To validate the predicted entity for each entity type, we provide the curated entity type definition as additional context in the format "**Context(definition) - Sequence**" as input to the boolq based text generation model. We also provide question to this model in the format, "**Does e_k follow the context mentioned?**" where e_k is the predicted entity. The fine-tuned model responds with either a "yes" or "no" answer, allowing us to retain the candidates which confine to the business contexts.

Since the fine tuned QA ensemble isn't powerful enough to provide only the appropriate entities due to absence of domain knowledge, and is frequently linked with false positives, this textual generation-based noise reduction helps us increase the coverage of predicted entities with minimum false positive rate.

2.5 Business rule based Candidate enrichment

In addition to the entities generated by the question answering models, we created a set of business rules to extract a few other entity types that could not be processed by question answering approaches due to lack of linguistic structure in the notes in which these entities appear. These entities include named entities that we have some prior knowledge about and would like the model to learn for generalisation purposes, such as Model Number. We generate annotated data for these labels using a collection of "seed entities" as the knowledge context. When we utilise this data to fine-tune the model, it learns the latent patterns in which these entities appear and predicts newer entities of this type that may appear in the future.

2.6 Aggregation of generated data

Once the data has been generated using these approaches, we pass it to a collator, which aggregates all of the various predictions in such a way that

– Each token has been tagged to only one single entity, since in our use case, it has been proven that there would be no scenario where one token would belong to more than one entity type.
– Each text sequence has a set of non-overlapping entities. The data generation pipeline tags multiple sub sequences of text to a particular entity type, there are cases where they might end up overlapping with each other. For example, in the sequence *The machine failed self-test during boot up* may have sequences such as *failed self-test* , *failed self-test during boot up* tagged to the entity type **Functional tests**. The collator looks at various factors such as model prediction

probability, length of sequence tagged and linguistic properties such as noun chunks to choose the most relevant sub sequence among these sequences.

2.7 Explanation and breakup of tagged data numbers

At the end of this module, for the unstructured agent note inputs, we have credible tagged data which can now be fed into our named entity fine tuner. The distribution of the tagged data is as depicted in 1 :

Table 1. Data distribution

Entity type	Number of samples	Percentage of contribution
O	111662	79.69%
I-Test	17565	12.53%
B-Test	5539	3.95%
I-Replaced Parts	1895	1.35%
B-Replaced Parts	1057	0.75%
B-model Number	624	0.45%
I-Amplitude	503	0.36%
B-Frequency	474	0.34%
B-Amplitude	397	0.28%
I-Frequency	334	0.24%
B-error code	35	0.02%
I-error code	31	0.02%
I-model Number	13	0.01%

3 Model Description

3.1 Model Architecture

The basic architecture we use for model training is depicted in 2

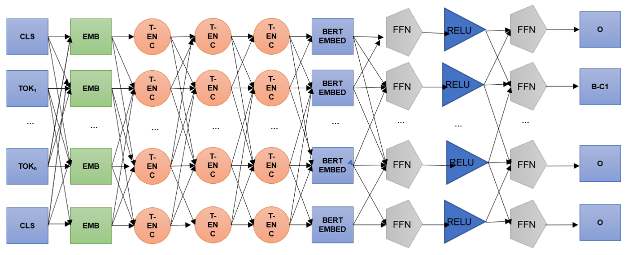

Fig. 2. BERT/ELECTRA/RoBERTa based NER model architecture

Based on the experiments we performed, we chose to fine-tune our NER task on pre-trained architecture such as BERT, RoBERTa and ELECTRA and compare the performance that was best suited for our use case. From the generated data, we provide tokens in the sequence and named entity types with the B and I tags (to indicate if the token is a start or rest of the named entity) as input to the model. So we define a set of tokens $C = \{c_j\}$ i.e. $[CLS]\,, c_1\,, c_2, ..., c_n\,[SEP]$ as input to the pretrained model where n is the number of tokens present in the text. We provide named entity tags such as $O, B - e_k, I - e_k, ..., O, O$ as output for our model. For the input text, the model predicts a tag for each token at the end of the output layer.

3.2 Contextualization

We chose transformer based architectures such as BERT, RoBERTa and ELECTRA to find the ideal model for our use case. We modified the architectures by adding a contextualisation layer in terms of two fully connected layers. We chose fully connected layers since a we intend to learn features from all the combinations of the embedding and to learn maximum information with the limited data made available to us. To handle the interaction effects and capture non-linearity of the data in a better way, we add a ReLU unit at the end of each fully connected layer. We take the output of these fully connected layers and feed it to the final feed-forward output layer to predict the entity tags for each token.

3.3 Training and Testing

During training, the tokens for each sequence X (generated based on context, definitions and entity types) have annotations of e_k where k is the number of entity types for each token. We calculate categorical cross entropy loss for each token as

$$CE_k = -\sum_{i=1}^{C} t_i log(p_i) \tag{1}$$

where C is the total number of entity classes, t_i is a binary indicator if class label c is a correct prediction for the token k and p_i denote the predicted probability of token k belonging to class c.

The model is trained for 15 epochs with learning rate of $5e^{-5}$ and validated using f1-score on a hold-out sample. We trained the same architecture with various pre-trained transformer architecture such as BERT, RoBERTa and ELECTRA. The models were trained with same architecture so that their performances can be compared.

During inference, the text is passed as tokens and the tags which start with B and continue till I are considered as a single entity and validated accordingly. In practice, we deployed the best performing model among the three architectures to suggest named entities to technicians.

4 Experiments and data

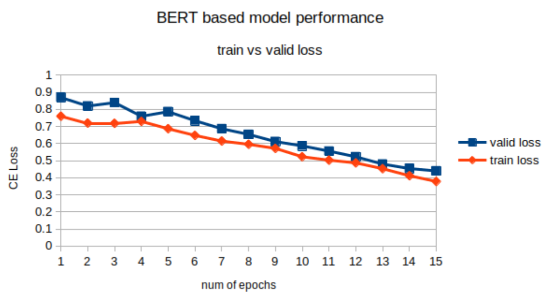

Fig. 3. train and val loss for BERT based NER

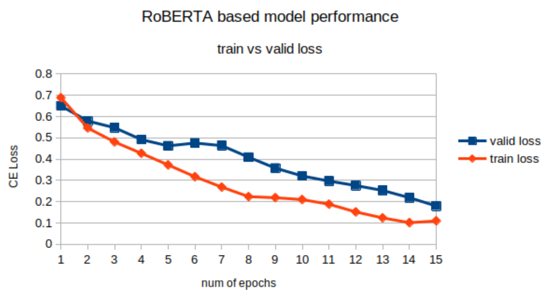

Fig. 4. train and val loss for RoBERTa based NER

4.1 Comparison of performance metrics

The following table 2 shows various performance metrics such as precision, recall and f1-score for the final fine-tuned models based on the various architectures discussed.

We use a learning rate of $5e^{-5}$ for all our experiments. The maximum sequence length was defined as 128 based on what we saw in the training data. This was based on the average sequence length of the technician notes that we encounter. During inference, if we face longer sequence, we break the text into logical chunks of 128 token-long sequences and process that for testing. We use the metrics train loss, validation loss, precision, f1-score for evaluation of model performance.

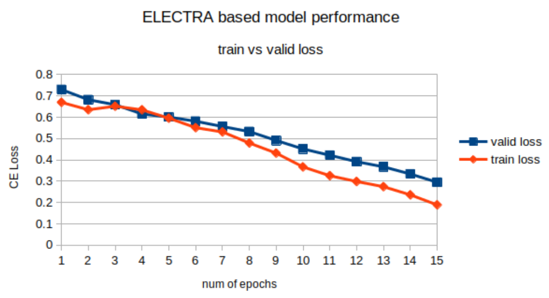

Fig. 5. train and val loss for ELECTRA based NER

	ELECTRA		BERT		RoBERTa	
	TRAIN	VALID	TRAIN	VALID	TRAIN	VALID
LOSS	0.19	0.30	0.38	0.44	0.11	0.18
PRECISION	0.72	0.71	0.65	0.56	0.81	0.80
RECALL	0.77	0.70	0.63	0.56	0.81	0.78
F1-SCORE	0.74	0.69	0.64	0.55	0.81	0.79

Table 2. Overall Performance and loss report

4.2 Comparison of train and validation loss

The train and validation loss for the models fine tuned based on BERT, RoBERTa and ELECTRA based architectures are depicted in 3 to 5 . From the figures we can see that the RoBERTa model has got the least validation and train loss and this is also reflected in the performance of the model for individual entity tags as well.

4.3 Class level performance comparison

The class level metrics for the models were depicted in 3. We compare performance metrics such as train and validation precision, recall and f1. To maintain equal significance to all the classes, including the imbalanced ones, we use macro averaging based F1 measure.

Macro F1 measure is calculated as follows

$$\begin{gathered} Macro\,F1score = \frac{1}{C}\sum_{i=1}^{N} f1_i \\ where \\ f1 = \frac{(2 * precision * recall)}{(precision + recall)} \end{gathered} \tag{2}$$

4.4 Inference

The model trained based on the RoBERTa based embeddings performed consistently on all the classes in both train and validation data. This can also be infered

		ELECTRA						
		Amplitude	**Frequency**	**Replaced Parts**	**Test**	**error code**	**model number**	**Overall**
TRAIN	**PRECISION**	0.83	0.64	0.71	0.67	0.67	0.81	**0.72**
	RECALL	0.60	0.64	0.79	0.82	0.81	0.99	**0.77**
	F1-SCORE	0.69	0.64	0.75	0.74	0.73	0.89	**0.74**
VALID	**PRECISION**	0.84	0.58	0.58	0.59	0.83	0.83	**0.71**
	RECALL	0.52	0.70	0.69	0.66	0.62	1.00	**0.70**
	F1-SCORE	0.64	0.64	0.63	0.62	0.71	0.91	**0.69**

		BERT						
		Amplitude	**Frequency**	**Replaced Parts**	**Test**	**error code**	**model number**	**Overall**
TRAIN	**PRECISION**	0.64	0.59	0.70	0.69	0.61	0.70	**0.65**
	RECALL	0.48	0.58	0.60	0.61	0.55	0.98	**0.63**
	F1-SCORE	0.55	0.58	0.64	0.65	0.58	0.82	**0.64**
VALID	**PRECISION**	0.64	0.32	0.57	0.49	0.58	0.76	**0.56**
	RECALL	0.49	0.31	0.54	0.42	0.59	0.99	**0.56**
	F1-SCORE	0.56	0.31	0.56	0.45	0.58	0.86	**0.55**

		RoBERTa						
		Amplitude	Frequency	Replaced Parts	Test	error code	model number	**Overall**
TRAIN	**PRECISION**	1.00	0.59	0.71	0.76	0.85	0.93	**0.81**
	RECALL	1.00	0.67	0.73	0.75	0.82	0.99	**0.83**
	F1-SCORE	1.00	0.63	0.72	0.76	0.83	0.96	**0.82**
VALID	**PRECISION**	1.00	0.67	0.66	0.74	0.77	0.95	**0.80**
	RECALL	1.00	0.65	0.62	0.69	0.75	0.98	**0.78**
	F1-SCORE	1.00	0.66	0.64	0.71	0.76	0.96	**0.79**

Table 3. Class level performance metrics

from the performance charts and tables. However, the BERT and ELECTRA based models were not able to generalize well on the validation set and hence do not perform as well as the RoBERTa based model. So, for final inference in the product, we used the RoBERTa based model for all the entities.

5 Conclusion and Future Work

For the task of NER, we adopt various pretrained models (BERT, RoBERTa) in this paper. First, we concentrate on data generation utilising unsupervised methods and sequence-to-sequence models. Then, for these entity types, we utilise certain business definitions to eliminate the noisy labels that are generated, and came up with empirical rules iteratively to further minimise the noise in the data. This yields a final annotated dataset that could be utilised in any of the NER training architectures that are based on pre-trained models. Furthermore, we show that RoBERTa based models are better suited to our NER task.

For the purpose of this study, we focused on data quality and quantity and utilized data generation and augmentation techniques to arrive at the best possible training data from unstructured data. In the future, we would be concentrating on few shot learning methodologies that would improve the model's performance. We also want to incorporate active learning based mechanisms to improve model's performance as discussed in [20].

References

1. Paul Thompson and Christopher Dozier. Name searching and information retrieval. 12 2002.
2. Radu Florian, Abe Ittycheriah, Hongyan Jing, and Tong Zhang. Named entity recognition through classifier combination. *Proceedings of CoNLL-2003*, 03 2004.
3. Martin Hassel. Exploitation of named entities in automatic text summarization for swedish. 2003.
4. Kamal Raj Kanakarajan, Bhuvana Kundumani, and Malaikannan Sankarasubbu. Bioelectra:pretrained biomedical text encoder using discriminators. In *BIONLP*, 2021.
5. Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly available clinical BERT embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, 2019.
6. Service Intelligence Product, Neuron7.ai. https://www.neuron7.ai/.
7. Hye-Jeong Song, Byeong-Cheol Jo, Chan Park, Jong-Dae Kim, and Yu-Seop Kim. Comparison of named entity recognition methodologies in biomedical documents. *BioMedical Engineering OnLine*, 17, 11 2018.
8. Rafiullah Momand, Shakirullah Waseeb, and Ahmad Latif Rai. A comparative study of dictionary-based and machine learning-based named entity recognition in pashto. pages 96–101, 12 2020.
9. Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. 01 2014.
10. A. Akbik, Tanja Bergmann, Duncan A. J. Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. Flair: An easy-to-use framework for state-of-the-art nlp. In *NAACL*, 2019.
11. Sudha Morwal, Nusrat Jahan, and Deepti Chopra. Named entity recognition using hidden markov model (hmm). *International Journal on Natural Language Computing*, 1:15–23, 12 2012.
12. Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *CoRR*, abs/1508.01991, 2015.
13. Jing li, Aixin Sun, Ray Han, and Chenliang Li. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, PP:1–1, 03 2020.
14. Ji Lee, Franck Dernoncourt, and Peter Szolovits. Transfer learning for named-entity recognition with neural networks. 05 2017.
15. Dana Ruiter, Dietrich Klakow, Josef Genabith, and Cristina España-Bonet. Integrating unsupervised data generation into self-supervised neural machine translation for low-resource languages. 07 2021.
16. Omid Jafari, Parth Nagarkar, Bhagwan Thatte, and Carl Ingram. Satellitener: An effective named entity recognition model for the satellite domain. pages 100–107, 01 2020.
17. Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018.
18. Yuwen Zhang. Bert for question answering on squad 2 . 0. 2019.
19. Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.
20. G Abinaya, Gyan Ranjan, and P Aswin Karthik. Continuous learning mechanism of nlu-ml models boosted by human feedback. In *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, pages 1–6, 2019.