A Custom Named Entity Recognition System for Customer Complaint Text in Consumer Electronics

Rithun S, Idhaya Venthan S, Tharun Gandhi P, Amala Balakrishnan Gesulan, Poli Vardhini Reddy, Praveen Kumar Reddy Maddikunta

School of Computer Science Engineering and Information Systems, Vellore Institute of Technology, Vellore, Tamil Nadu, India

Abstract

Named Entity Recognition is an important task of NLP that involves identification and classification of named entities in the text. Although till now, much study on NER has been done in many domains, it is still a challenging task when performed on new domains where annotated data is less. In this paper, we describe an NER system for the consumer electronics complaint domain that addresses the challenge. Our system uses RegEx and supervised methods to come up with a model that would be totally or near-human-annotation-free. On the benchmark dataset of complaints, our system evaluated well against baseline scores, achieving competitive results. To the best of our knowledge, this is the very first study aimed at developing an NER system targeting Consumer Electronics Complaints. Our findings demonstrate the potential of the proposed system in consumer electronics for NER and set the future direction of research in this area.

Keywords: Named Entity Recognition, NER, SpaCy, RegEx, E-Commerce, Machine Learning

1. Introduction:

Since the last decade, we have been able to digitalize a large number of handwritten records, such as FIRs, property details, health records, and citizenship information. However, how are we going to classify them and use them when we need them? This question arises because the digitized data can be both structured and unstructured. Sorting the unstructured data is a challenge for humans but not for machines unless we train them. We humans know how to sort for limited data but how are we going to do that for big data it is time-consuming. We can perform this task more efficiently by training systems to process data in their own language through natural language processing (NLP) [1]. Archaean Goyal discusses the application of NLP to handle large datasets, particularly for processing Tamil documents. With the increasing volumes of data, it's crucial to develop new technologies like NLP to manage and utilize this data effectively [2].

Once NER is implemented, we need to understand what entities are. Generally, entities are noun phrases, with named entities being the most common. Named entities can include names of people, companies, locations, and more. These entities are helpful for tasks such as text summarization [3].as years passed we could able to develop new entities for domain specific needs say for medical or legal sectors [4] but since they are to be domain specific we could not get them with default pre trained spacy rather we should train or fine tune the model with respect to our needs just like this, researchers have proposed a hybrid NER model mainly designed for aviation industry by obtaining high accuracy through identifying and classifying entities based on civil aviation they obtained high accuracy through rule based and supervised learning approaches [5].

let’s see a simple example for understanding entities:

- Input: Elon musk is going to the Himalayas on 22-07-2024 for his vacation in India with his children.
- Output: ('Elon Musk', ‘person’), ('Himalayas', 'LOC'), ('22-07-2024', 'DATE'), ('India', 'GPE')

In this example, “Elon musk” is the entity, and “person” is the label. In same way we can describe the rest of the entities.

In the simple terms, NLP facilitates human-computer interaction. SpaCy, an NLP library, includes a pre-trained model that identifies entities based on capitalization, prefixes, suffixes, parts of speech, and more. However, pre-trained NER models are not always accurate. For instance, without capitalization, these models struggle to identify entities. However, since Natural language processing is highly contextual it is ambiguous, especially in language-specific domains Srinivasan and Rajalatha has worked on extracting entities from Tamil documents using Rule based approach along with Naive Bayes they used Regex patterns, Morphological and Contextual Feature Extraction [6].some companies use native language branding to mark their cultural relevance and sometimes misspelling may occur with dealers who post on E-commerce sites and E-commerce domains are wide we need domain specific knowledge to show best results.

The pre-trained spaCy model is not entirely accurate, which is why we use hybrid learning approaches that combine rule-based methods and machine learning techniques. Since NLP is in its nascent stage, and research are applying Named Entity Recognition (NER) to domain-specific studies such as aviation, engineering tools and automated NER from Tamil documents. Given its domain specificity, we can use spaCy’s pre-trained model in conjunction with hybrid learning approaches for better accuracy compared to using only pre-trained model. Learning-based approaches can be classified into three categories: supervised, unsupervised, and semi-supervised.

Rule-based approaches are domain-specific, requiring the definition of a set of rules. This process is expensive, time-consuming, and demands a lot of human effort. However, as time has passed, many machine learning algorithms have been developed in the industry, allowing us to use learning-based approaches more effectively.

This model was created out of frustration in personal filing of a complaint for a laptop device. The system available was inefficient, cumulative, and had no categorization for specific issues like overheating. Instead, it handled the complaints in a general way, requiring the customer service provider to follow up for more details. Since my purchases were online, I could only lodge complaints via the website, which usually includes a long delay before the complaint reaches the proper service provider. This inefficiency showed a deficiency in the system that a model designed to fast-track the process of complaint could very well address. It was supposed to find out a way in which the complaints could automatically get filtered and categorized and then let them be routed to the appropriate brand service provider with highlighted issues. In doing so, this can speed up the resolution time and offer better customer satisfaction, as very less time and effort are taken to take up and fix the issues.

The rest of the paper is organized as follows. Section 2 discusses the related works, which is followed by the methodology in Section 3. Section 4 elaborates the results that were obtained. Finally, Section 5 discusses the conclusion and future directions.

2. Related works

Siddharth Ruppani et.al, in their paper on role matching using machine learning, says that Support Vector Machines (SVMs) work well compared to other text classification methods [7] since SVMs are much more robust, even while handling incomplete or grammatically incorrect sentences.

Application of NER to solve product matching is critical in the scaling up. We are focusing more on advancing our E-commerce by using product titles and descriptions for product matching [8] rather than just gathering the necessary information from text, which is what Named Entity Recognition does. A good NER needs the right rule-based and machine learning algorithms. This approach to product matching further increases the importance of NER in E-commerce.

Large performance gains in the paper proposed by X. Yang et.al [9] were obtained by adding character-level GRUs and gazetteers to the base model; this went as high as 91.20 in F1 score. This result indicates that traditional feature engineering still has a place in the improvement of state-of-the-art NER systems using advanced neural architectures.

The paper from Emma Strubell et.al [10]. introduced dilated convolutions in the NER model, which efficiently processes larger input windows without losing too much accuracy. Their approach retained near state-of-the-art results, substantially improving computational efficiency for results—an important factor when dealing with real-world applications.

A paper proposed by Popovski et.al [11] applied a rule-based approach, enhanced with semantic information and POS-tagging, to the FoodIE method oriented to food recipe names. It thus showed the strength of hybrid methods within specialized domains. Another example is the OGER++ tool, proposed by Furrer, combining dictionary-based entries and text disambiguation to extract and link biomedical entities efficiently, which shows the utility of hybrid systems in domain-specific NER tasks.

Another highly influential work concerning this topic was proposed by Kubala et.al [12], where these authors used an HMM for the extraction of named entities from speech transcription systems. In this case, their combined score in precision and recall during the MUC-7 competition emerged to be 93.9%, thus putting into evidence the efficiency of the method in NER tasks.

Deep Cross-platform product matching in E-commerce by Juan Li et.al [13], proposed in their paper a TMM (Title Matching Model) that represents the use of bidirectional long- and short-term memory neural networks in specifying semantic representations of titles and interaction methods. It includes convolutional and pooling layers used in the creation of instance-level representation and aggregation into field-level representations of relevance for entity extraction in product descriptions.

Item Matching Model in E-commerce by Olga Cherednichenko et.al [14] refers to the development of E-commerce and a consequent trend which has affected item searching. Thus, it covers the development of an item-matching model, semantic closeness of text descriptions, and practical implications for improving processes of item searching on the platform of E-commerce. The paper clearly articulates the role of machine learning algorithms in detecting relevance between items and thus bettering user experience in E-commerce.

These related works display a very wide variety of methodologies and innovations in the area of NER that can be used in supporting the development and subsequently evaluating the quality and performance of our model.

3. Methodology

In this section, we describe our methodology for constructing an NER model for Consumer Electronics with SpaCy. The reason for choosing SpaCy is that this library has been very accurate in any NER task and shows leading performance benchmarks, mostly ranking above other popular libraries. Moreover, SpaCy is designed to support seamless deployment, which makes it easy to integrate our NER model into real-world applications. It is owing to this user-friendly API that extensive customization in SpaCy is further possible development of a tokenizer and integration of our heuristics and rules to improve the precision of the system.

The tasks that we are going to perform include identifying and classifying entities such as Brand, Year, Display Size, Processor, Storage, Color, OS, and Issue. Since a complete annotated dataset has been provided pertaining to consumer electronics, we followed a structured training methodology. First, using Regular Expressions with a pre-trained SpaCy model, we obtained the preliminary labels for our target entities. These labels will become the basis for the training of our custom SpaCy NER model. Figure 2 illustrates our proposed methodology, outlining key steps and processes in our approach.

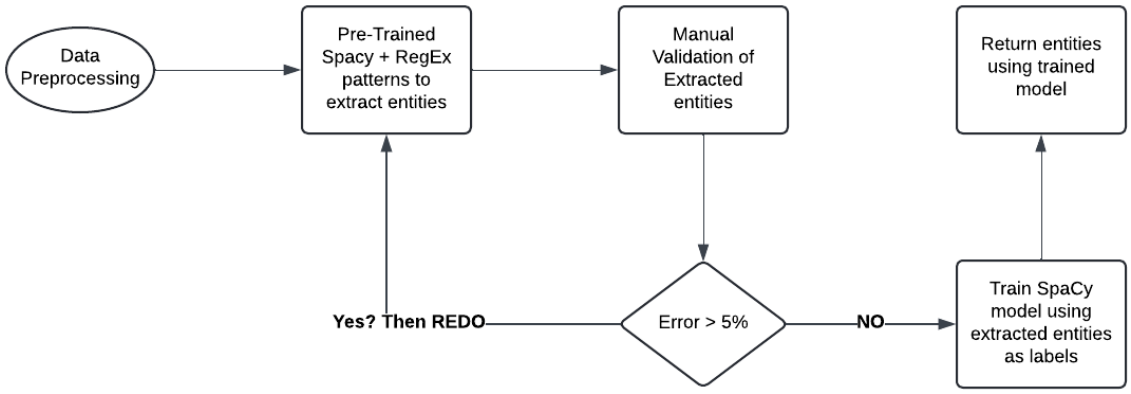

Figure 1. Proposed Methodology

3.1. SpaCy + RegEx

In the first line of our NER approach, extraction of entities from product data and customer complaints is combined with RegEx and the pre-trained SpaCy NER. The approach is further used for extracting entities, and also the outputs of this model are used as labels in training a SpaCy NER model on our entities of interest. In the case when no annotated training data is available, then, with the help of simple heuristics, it is possible to generate labels effectively to train an NER system for specific entities.

RegEx is a sequence of characters that defines a search pattern. We are in a position to define appropriate patterns for identifying the Entities of Interest from the text data. For instance, to match specific attributes, such as processor models, we used regular expressions to match patterns like "Intel Core i7-12700H" or "AMD Ryzen 9 5900HX". Presently, in this research, a RegEx-based approach is followed to extract entities corresponding to Brand, Product Line, Processor, Issue, and Possible Fix. This set of heuristics and rules are dataset-specific and applied to the text in order to identify this set. The major advantage is the outright speed and ease of deriving labels through RegEx model. There will be no need to perform any manual annotation of data, which can be a very time-consuming and labor-intensive activity. Further, RegEx models are scalable; in fact, we can use the same patterns against huge amounts of text data that are very hard to annotate manually. First, we create a SpaCy document from the product and complaint text. Second, we utilize the re-tokenize function of SpaCy, which combines possible multi-token entities into a single token. Third, we loop over each token in the document. Fourth, we use a set of context rules and the surrounding words to find certain entities that are being extracted by us in two ways. The first implementation is accomplished by leveraging the predicting entities by pre-trained SpaCy NER model. We have a set of context heuristics to try and label a token to a certain entity. Also, we maintain a series of rules where we block SpaCy from classifying irrelevant tokens-generic terms that don't pertain to our attributes of interest. We noticed that the default SpaCy tokenizer was not really taking care of the words with hyphens in them, such as "Intel Core-i7-12700H", by separation into individual tokens. Therefore, we came up with a custom SpaCy tokenizer that enables us to keep the hyphenated words as a single token. We also rid some of the suffix rules, such as breaking up letters following a number like in the case of "Ryzen-9"; this would be divided into two different tokens by SpaCy. This preprocessing step increases the accuracy of our NER system by ensuring that entities with hyphenated words or suffixes are not missed. Having developed our RegEx + pre-trained SpaCy NER approach, we applied the pipeline to the whole processed dataset from the collected product descriptions and customer complaints. This was later used to extract and classify responses in a diverse range of text data. When taking this route, we could generate a large dataset of labels by which to train the SpaCy NER model. Although RegEx helps to generate labels, in a way, one must also point to its limitations. The RegEx models are not always accurate, and sometimes the patterns fail to capture all possible variations of the entities of interest. RegEx models do not identify entities of interest because of format variations in products' names or attributes. Therefore, the quality of the labels created by the RegEx model needs to be validated as true labels before being used as training data for the NER system. Therefore, several strategies were followed to overcome this limitation concerning the validation of the quality of labels generated by our RegEx model. And this set up an iterative approach for us to make sure that the label data produced by our RegEx model is of high quality and fit for training the NER system.

3.2. Extracting “Issue” as Entities

First, we load complaints from the JSON file named `complaints.json`. Each complaint should have a key that is a string with a "complaint" key. These are stored in a list for further processing.

Our code extracts entities based on regex patterns. It defines the `ISSUE` as entities. The `ISSUE` pattern here is very elaborate, covering most the problems that might ever be reported by users, such as "screen flickers," "overheating," and "random shutdowns" and more. After defining these patterns, they are then compiled into regex objects to enhance performance during extraction.

The `extract_entities` function processes each complaint text through the spaCy language model to create a document object, then applies the precompiled regex patterns to the text in order to identify matches. This function will then create a span object via the described math using the 'char_span' method of spaCy, labeling the matched text with the proper entity type. In case of the span being valid, it would be added in a list of entities.

The program iterates through every complaint, passing the text to extract entities with the 'extract_entities' function. Store results in the new list formatted to be used with spaCy's NER training. Each entry in such a list contains the complaint text and a list of entities, which are identified by their start and end character positions and their label.

Finally, the complaints are processed and extracted entities are written out to a new JSON file, such as `processed_complaints.json`. This JSON file contains all the complaint text with their extracted entities, formatted in a way that will be readily usable for training an NER model. For instance, a complaint such as "My HP Pavilion with an AMD Ryzen 5 processor is overheating after just 30 minutes of use" would have entities labeled as "HP" for the brand, "AMD Ryzen 5" for the processor, and "overheating" for the problem. This enables a structured approach to ensure that every complaint has relevant information annotated, hence providing a very strong dataset for training a spaCy NER model.

3.3. Training SpaCy

Annotated training data with these labels formed the basis for our effective training of the custom SpaCy NER model. We have achieved such labeling with the help of a combination of regular expressions and a pre-trained SpaCy model. The training data had been pre-formatted, hence having the entity annotations and token indices; it just needed to be converted into a format for which SpaCy models are trained.

We started off by making a '.spacy' file with the training data in the correct format.We are using the class 'DocBin' from the SpaCy library. We have managed to create a SpaCy Docs binary file. Creating involved iterating over each data in the training dataset to create tuples for each entity with start and end indices together with the label for the entity. These tuples were then put into the 'Doc' object using the 'set_ents' method. Finally, the 'Doc' object was serialized to the '.spacy' file by the 'to_disk' method from class 'DocBin'. The data was finally split into a training set and a validation set after it had been turned into SpaCy format. It is in this regard that this partitioning strategy set aside 70% for training and 30% for validation to ensure the model evaluation would be stringent enough.

We then trained the custom NER using 'train' from SpaCy. In our custom NER pipeline, we applied the pre-trained model 'en_core_web_lg' as our base model. We configured our training through config.cfg with hyper-parameters such as a batch size of 100 and a learning rate of 0.001 for stability and the speed of convergence. Again here, we can see the frequency of evaluation, which is set to 200, providing the model with regular performance checks during training. Another critical parameter in our configuration file, 'config.cfg', very critical to the robustness of the model against overfitting, is the Dropout Rate and L2 Regularization. The Dropout rate enhances the generalization part of the model by reducing how dependent the model is on some particular neuron during the training phase. Similarly, L2 regularization penalizes large weights, promoting simpler models that won't overfit the training data. These configurations were critical in structuring the training dynamics that helped develop a high-performing NER model.

3.3.1. Evaluation Metrics

Precision is an accuracy measure for positive predictions made by the model. It is defined as the number of correctly recognized NEs divided by the total number of NEs retrieved by the model. High precision means the model has a low false positive rate; that's to say, it only returns those that are correct entities.

$$\text{Precision} = \frac{(\text{number of correctly recognized NEs})}{(\text{Total number of NEs retrieved by the model})} \quad \text{[EQUATION 1]}$$

Recall measures the model's ability to retrieve all instances of interest from a dataset. Mathematically, recall is the number of correctly recognized NEs divided by the total number of actual NEs in the dataset. High recall would imply that the model has a low false-negative rate since it retrieves nearly all relevant entities existing in the text.

$$\text{Recall} = \frac{(number\ of\ correctly\ recognized\ NEs)}{(Total\ number\ of\ NEs)} \quad \text{[EQUATION 2]}$$

The F1 Score is the harmonic mean of Precision and Recall, and it provides a single measure that balances the two concerns. The F1 score is very useful when you need a balance between precision and recall, or when there's an uneven class distribution.

$$\text{F1 Score} = \frac{(2*p*r)}{(p+r)} \quad \text{[EQUATION 3]}$$

For example, on the document level, the custom NER model performed far above the performance metrics compared to the pre-trained SpaCy model. In fact, precision, recall, and F1 score for the custom model were all above 93%, thus proving that it is much better at correctly identifying and classifying entities compared to the baseline model, which had a lower precision, recall, and F1 score of 24.43, 40.94, and 30.60, respectively.

Aided by high scores on these metrics, the custom model has thus been an effective means to correctly extract relevant entities while keeping false positives and negatives at bay for guaranteed real-world performance.

Table 1. Pre-trained SpaCy vs Our Model

NER Model	Precision	Recall	F1 Score
Pre-trained NER	24.43	40.94	30.60
Custom-built NER	93.79	93.67	93.73

4. Results and Discussion

A number of steps have been taken to ensure top-grade data collection during the process of developing a high-quality dataset for training a named entity recognition model using spacy. Leading E-commerce websites such as Amazon and Flipkart were used as main sources, since they hold huge lists of products with rich descriptions. We focused on products like laptops, smartphones, smart watches, and televisions. Data collection was done very meticulously.

Data Collection and Preparation

We prepared the product data by manually browsing through the selected categories in Amazon and Flipkart. Such a manual process ensured the accuracy and appropriateness of the data. We copied the relevant data of the products and stored them systematically in JSON files. Every product entry in the JSON file consisted of attributes such as brand, manufacturing year, display size, processor, storage, color, OS, and issue. The dataset was cleaned properly to remove inconsistencies and unwanted data. Standard formats were created for attributes, and special characters like "\", "(", and ")" had been removed for convenience. Cross-checking among different platforms was done during cleaning for the accuracy of data. This was a very important step in the formation of a strong dataset because it checked the accuracy of every product.

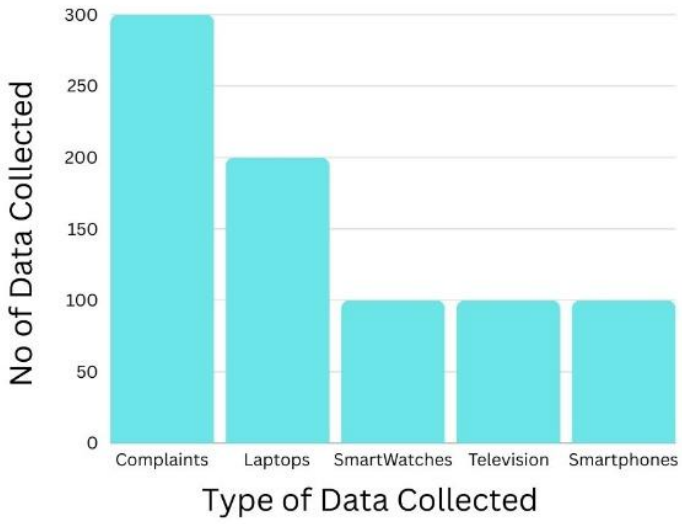

Figure 2. Collected Data

Dataset Composition

We gathered 500 points of data regarding electronic equipment: 200 on laptops, 100 on smart watches, 100 on smart phones, and 100 on televisions. Added to this were 300 customer complaints that were sourced through forms and customer reviews, thereby making it 800 entries in total. Data from customer complaints emphasized the problem faced by customers with their products. An example of a laptop entry in our dataset is: "HP Pavilion, 2022 – AMD Ryzen 5 5600H, 56WHrs Battery, 16 GB, 1 TB SSD, Windows 11 Home, Intel Iris Xe Graphics, Ultrabook, 13.3 Inch, Silver, 1.35 Kg." A corresponding consumer complaint is: "My HP Pavilion with an AMD Ryzen 5 processor is overheating after just 30 minutes of use."

Customer Complaints Data

Customers' complaints were scrapped from online forums, review sections on E-commerce websites, and feedback portals of customers. Sources relevant to this scanning exercise are those where customers most frequent and discuss issues related to their purchases. There were 300 complaints collected and filtered, providing a substantial data set for analysis. A comprehensive approach in the collection and cleaning of data ensures that the reliability and uniformity of the dataset for the training of an accurate and effective NER model using spaCy is thoroughly supported.

Training Results

After validating the RegEx model's output, we used 300 data points of complaints and 500 datapoints of products in training our SpaCy model. The data is split into 610 training data points and 190 validation data points. The reason we split the data like this is because we wanted to make sure that 70% of data points in both complaints and products are used to train and 30% of data points in validation. We have trained on all entities of interest, which are: BRAND, YEAR, PROCESSOR, STORAGE, OS, DISPLAY SIZE, COLOR and ISSUE.

Table 1 compares the performance of our proposed model against a baseline SpaCy pre-trained model in terms of precision, recall, and F1 score evaluation metrics. According to the table, for our proposed model, all scores are considerably higher than those for the baseline model, thus establishing its better performance in the task of identifying relevant entities and their relations in consumer electronics complaints.

Our model combined RegEx patterns with a pre-trained SpaCy NER model. At the same time, the combination aided by a custom-made tokenizer and heuristics-based labeling improved the accuracy to a large extent in entity extraction. The RegEx model efficiently generated labels, thereby eliminating the need for manual annotation.

We cantered on a variety of different hyper-parameters and features in SpaCy. At the very end, we achieved an F1-score of 93%, both for validation and test sets, considering all entities. This demonstrates the strength of performance in the proper identification and classification of entities in complaint texts.

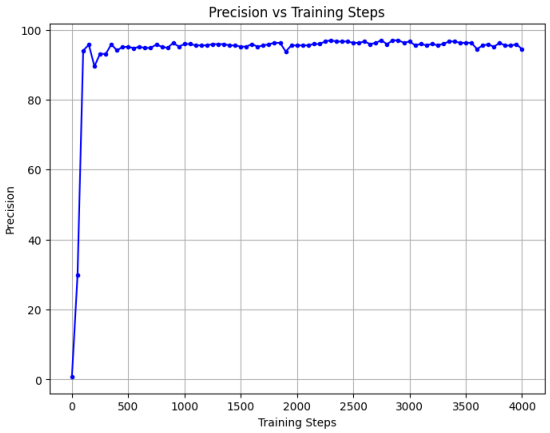

Figure 3 (a) Precision vs Training Steps

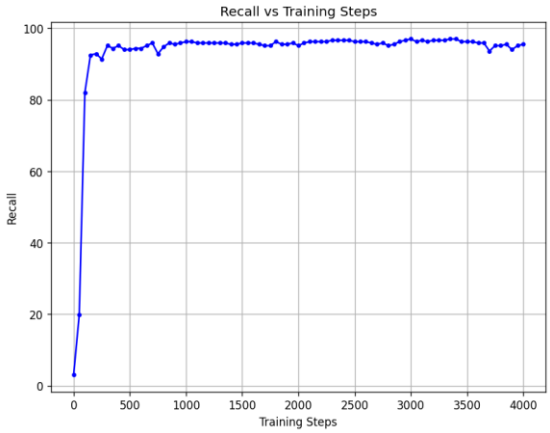

Figure 3 (b) Recall vs Training Steps

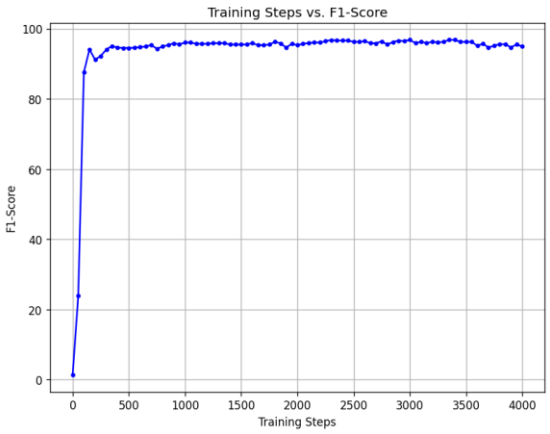

Figure 3 (c) F1-Score vs Training Steps

Figure 3 (a) shows the model's precision, which is very low at the beginning, about 0.86 at step 0, but it improves very fast in these first few steps of training and reaches a value of about 94% at the 150th step. Afterwards, the model precision improves further and stabilizes at about 96% from the 900th step onwards. Hence, the better the model learns from data, the more accurate it will be in predicting the correct named entities. As shown in the Figure 3(b) Recall starts very low as well, at about 3.00 in step 0, which means the model misses many true named entities. However, this value improves significantly in the first phases of training to about 93% at the 150th step and then goes on a slight rise to the end, stabilizing between 96%, matching precision, thus the model will be balanced to identify true positives. Figure 3 (c) highlights the F1 Score (that is, the harmonic mean of precision and recall) starts very close to 0.01 in early steps of training. This measure increases rapidly as precision and recall get better to about 94% at the 150th step and continues leveling out to approximately 96% by the 900th step, thus showing an overall improvement and stabilization of model performance.

It remains stable from about the 900th step because, by then, it would have captured most of the entities and labels that exist in the training data. Early in training, the model improves rapidly as it adjusts its parameters to make minimal mistakes. However, it slows down as it continues to train because, by then, it would have captured most of the key features necessary for making accurate predictions. By the 900th step, the learning curve of the model begins to flatten out. What this means is that further training brings incremental improvements only, and that the model will have reached a stability point where it will then be strong on the task.

Figure 4. NER Outputs of the models

A comparison of sample outputs given by our product only model against a pre-trained SpaCy model is presented in Figure 4. It shows the plain text of a product description and the text with entities annotated by our model and the pre-trained SpaCy model, respectively. The entities are

marked with coloured boxes, and the bold text to the right of an entity term represents the type of entity (such as BRAND, PROCESSOR, or STORAGE).

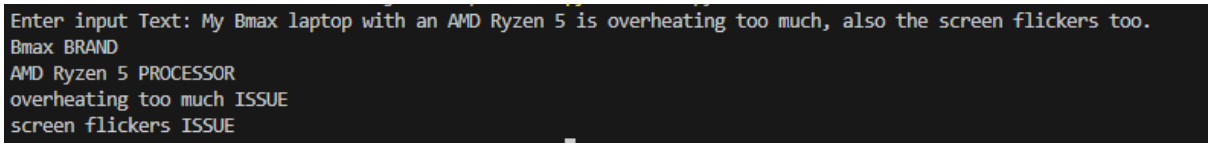

Figure 5. Final Model Output

The Figure 5 shows that the text "My Bmax laptop with an AMD Ryzen 5 is overheating too much. And the screen flickers too," the model identified "Bmax" as the BRAND, "AMD Ryzen 5" as the PROCESSOR, "overheating too much" as the ISSUE, and "screen flickers" as another ISSUE. This exemplifies the preciseness of the extraction of relevant entities by our custom-trained NER model.

5. Conclusion

The proposed solution is to develop a custom NER system that could smooth out the handling of customer complaints related to consumer electronics. By automatically parsing text complaints and classifying brand names, product specifications, and precise issues, and then forwarding them to the concerned service departments of a customer, the service teams could then focus on the solution unhindered by parsing the data. That's what the hybrid approach is all about: the regular expression-based model with a pre-trained SpaCy and added heuristics rules to custom tokenizers aims at achieving high precision and recall without a heavy dependence on human annotators. Data was scraped from E-commerce-related sites, cleaned, and used for training, with the result being an F1 score of 96% at 900th training step, which outperforms previous models. The final system performance showed a strong F1-score, since it was 93% for both the validation and test sets and thus was well applicable in actual real-life usage. Future work will include the application of the model to other domains and languages, the fine-tuning of domain-specific jargon, and ways of accounting for the structures of rare complaints to enhance model flexibility and precision.

6. Reference

1. Naseer, Salman, et al. "Named Entity Recognition (NER) in NLP Techniques, Tools Accuracy and Performance." Pakistan Journal of Multidisciplinary Research 2.2 (2021).

2. Archana Goyal, Vishal Gupta, and Manish Kumar, “Recent Named Entity Recognition and Classification techniques: A systematic review,”Computer Science Review.

3. Sarawagi, Sunita. "Information extraction." Foundations and Trends® in Databases 1.3 (2008).

4. I. Chalkidis, I. Androutsopoulos, and A. Michos. 2017. Extracting contract elements. In 16th International Conference on Artificial Intelligence and Law.

5. Bharathi, A., et al. "A hybrid named entity recognition system for aviation text." EAI Endorsed Transactions on Scalable Information Systems 11.1 (2024).

6. Srinivasan, R., and Chinnaudayar Navaneetha Subalalitha. "Automated named entity recognition from tamil documents." 2019 IEEE 1st international conference on energy, systems and information processing (ICESIP). IEEE, 2019.

7. Lin, Kai, et al. "Toy Matching in E-commerce."

8. H. Xu, W. Wang, X. Mao, X. Jiang, and M. Lan, “Scaling up open tagging from tens to thousands: Comprehension empowered attribute value extraction from product title,” in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.

9. X. Yang, J. Bian, Y. Gong, W.R. Hogan, Y. Wu, MADEx: a system for detecting medications, adverse drug events, and their relations from clinical notes, Drug Saf. 42 (2019).

10. Emma Strubell, Patrick Verga, David Belanger, and Andrew McCallum. Fast and accurate sequence labeling with iterated dilated convolutions. 2017.

11. Popovski, G., Kochev, S., Seljak, B. and Eftimov, T. (2019) FoodIE: A rule-based named-entity recognition method for food information extraction. In Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods (SCITEPRESS - Science and Technology Publications).

12. Kubala, F., Schwartz, R., Stone, R. and Weischedel, R. (1998) Named entity extraction from speech. InProceedings of DARPA Broadcast News Transcription and Understanding Workshop.

13. Juan Li, Zhicheng Dou, Yutao Zhu, Xiaochen Zuo, Ji-Rong Wen. Deep Cross-platform product matching in E-commerce.

14. Olga Cherednichenko, Oksana Ivashchenko, Ľuboš Cibák, Marcel Lincenyi. Item matching model in E-Commerece: How Users Benefit.

15. spaCy Documentation: spaCy 101: Everything you need to know (https://spacy.io/usage/spacy-101)

16. NER (Named Entity Recognition) (https://spacy.io/usage/linguistic-features#named-entities)

17. Training spaCy’s Statistical Models (https://spacy.io/usage/training)

18. spaCy [API Reference] (https://spacy.io/api)

19. "Natural Language Processing with Python and spaCy" by Yuli Vasiliev

20. "Mastering spaCy" by Duygu Altinok

21. SpaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing(https://www.aclweb.org/anthology/D18-1219/)

22. Improving Named Entity Recognition with CRF and Word Embeddings(https://arxiv.org/abs/1603.01360)