PII FILTER IMPLEMENTATION

By: Shreeyan Godey

What is Personally Identifiable Information (PII)?

Personally Identifiable Information (PII) includes any data that could potentially identify a specific individual. Examples include names, addresses, phone numbers, email addresses, social security numbers, etc.

Why is protecting PII important?

Protecting PII is crucial because it helps prevent identity theft, fraud, and unauthorized access to personal information. When PII falls into the wrong hands, it can be used for malicious purposes such as impersonation or financial scams.

The Goal

In this report, I will discuss identifying and filtering PII from large volumes of textual data with two main methods. I used two different sets of data to test and get scores for the precision, recall, and F1 score of all the methods.

What are the two main approaches evaluated in the report?

The report evaluates two main approaches for PII filtering:

- **AI-based Approach**: Uses GLiNER models, which employ natural language processing techniques.
- **Rule-based Approach**: Utilizes fixed filters including regular expressions, keyword matching, and NLP-based entity recognition using spacy.

Overview of the First Dataset

To test the rule-based filter and the ai-based filter, I made two different programs for each to test on two different datasets. A section of the first dataset is attached below:

```
[ {
  "tokenized_text": ["Mamadou", "Diop", ",", "a", "resident", "of",
"Dakar", ",", "is", "a", "regular", "at", "the", "'Club", "54'",
"nightclub", "located", "at","45", "Rue", "de", "la", "Liberté", ".",
"He", "usually", "arrives", "around", "midnight", "and", "often",
"leaves", "around", "3", "am", ".","Mamadou's","ID", "card", "number",
"is", "WS-123456789-1", ",", "and", "he","pays", "for", "his", "drinks",
"with", "a", "credit", "card", ":", "1234-5678-9012-3456", "."],
  "ner": [[0, 1, "person"], [14, 15, "nightclub"], [45, 45, "ID card
number"], [58, 58, "credit card number"]]
}, …
```

There are a few key things to note about the way this data is presented. First the text is tokenized so each word and punctuation mark is separated into individual elements within an array. Also, after the "tokenized_text" there is a field called "ner". This field contains the Named Entity Recognition data, which identifies and classifies possible PII within the text. The NER data is structured as an array of arrays, where each inner array represents a named entity with its start and end indices in the tokenized text, along with its classification.

This format enables efficient processing and analysis of the text's key components. It also can be used to grade the accuracy of my models by showing which words were supposed to be filtered. Now let's move on to the different ways I filtered this data. One other key thing to note is that a lot of the dataset includes different languages other than English.

AI-based Approach: GLiNER Models

I evaluated multiple versions of the GLiNER model in my program to see which would be the best one to use. These models include:

- **- GLiNER Base**
- **- GLiNER Multi**
- **- GLiNER Small (v1, v2, v2.1)**
- **- GLiNER Medium (v1, v2, v2.1)**
- **- GLiNER Large (v1, v2, v2.1)**
- **- GLiNER Multi v2.1**

GLiNER is a versatile tool for recognizing different types of entities in text. Unlike traditional Named Entity Recognition (NER) models limited to predefined entities, GLiNER can identify any type of entity, making it adaptable for various tasks and industries. It is also more efficient than large language models like GPT-3, providing similar flexibility without requiring a lot of computational resources. This makes GLiNER suitable for use in resource-constrained environments like this program. To get it in python all you need to write is:

```
!pip install gliner
```

How this code works is that it extracts the labels directly from the dataset and puts it into the model to filter it out. It first reads the input data, which contains tokenized text and corresponding entity labels. For each item in the dataset, the code retrieves the labels, feeds the text and labels into the GLiNER model to predict entity positions, and then replaces the identified entities in the text with a placeholder.

Another feature in GLiNER is called threshold. Threshold sets a confidence level that determines whether an entity is or isn't PII or whatever label you are trying to filter. When the model predicts an entity, it assigns a confidence score to that. If the score is below the threshold, the prediction is not counted. The lower your threshold is, the model will filter more entities even if it's not sure. If you increase it, it will only filter out entities it is very sure about. You can set it however you want based on what results you're looking for or fine-tune your F1-score by changing it.

Rule-based Approach: Fixed Filter

In addition to evaluating GLiNER models, I implemented a rule-based approach to entity recognition, which leverages traditional techniques such as regular expressions, keyword matching, and the spacy NLP. This technique has several components:

- It uses NLP-based entity recognition using the spacy library for names, places, organizations, and other things that spacy can recognize. For example here is one method for removing names from the text:

```
def remove_names(self, text):
        doc = self.nlp(text)
        for ent in doc.ents:
            if ent.label_ == "PERSON":
                text = text.replace(ent.text, "<FILTERED>")
        return text
```

- The code also uses regular expressions for things like dates, times, email addresses, URLs, and postal codes. This works because these types of entities in text will almost always follow a pattern. Here is an example of one of the methods that used regular expressions to filter out postal codes. It recognizes both five digit and nine digit postal codes and replaces them.

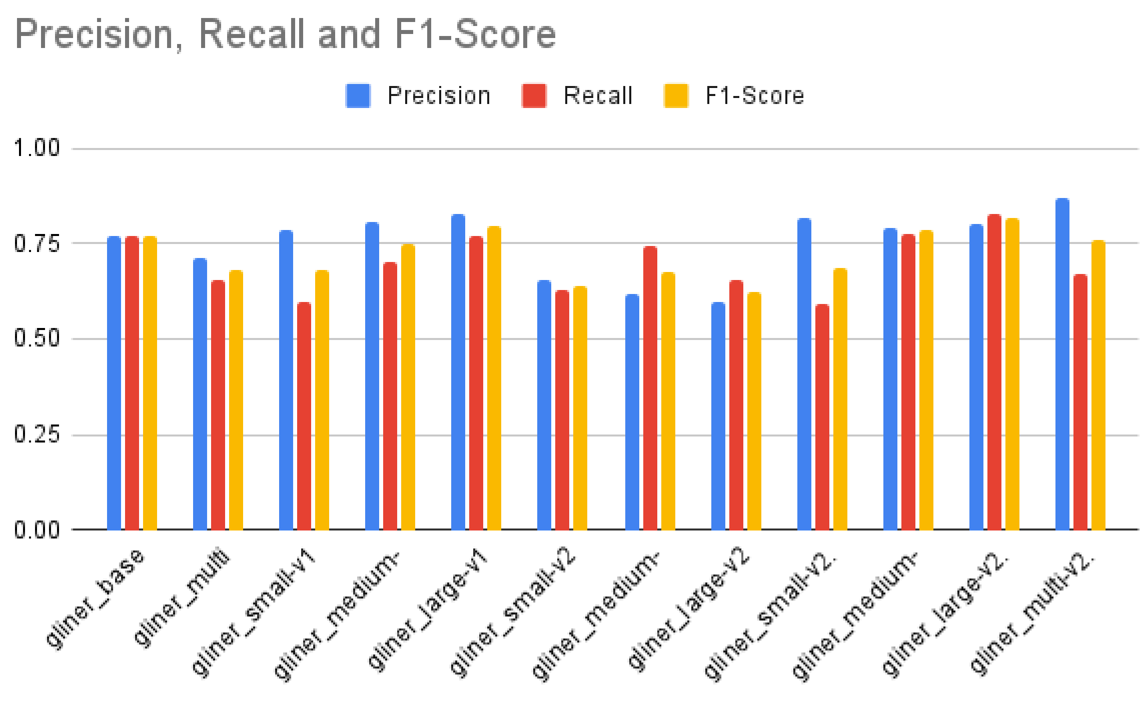
```
def remove_postal_codes(self, text):
        return re.sub(r'\b\d{5}(-\d{4})?\b', '<FILTERED>', text)
```

Results of the First Dataset

- Now that we have a quick overview of both the AI-based approach using GLiNER models and the rule-based approach using spacy works, let's discuss the results. For each method, I evaluated their ability to accurately identify and filter out PII from the dataset. The key metrics used for comparison include precision, recall, and F1-score, which are standard measures for evaluating the performance of entity recognition systems.

- **Precision** measures the accuracy of the entities identified by the model: it will go lower if the text that was filtered is not marked as PII, also known as a false positive.
- **Recall** measures the completeness: it will go lower if the model missed some text that is PII, also known as a false negative.
- The **F1-score** provides a balance between precision and recall, giving a single metric for comparison.

GLiNER Scores for the First Dataset

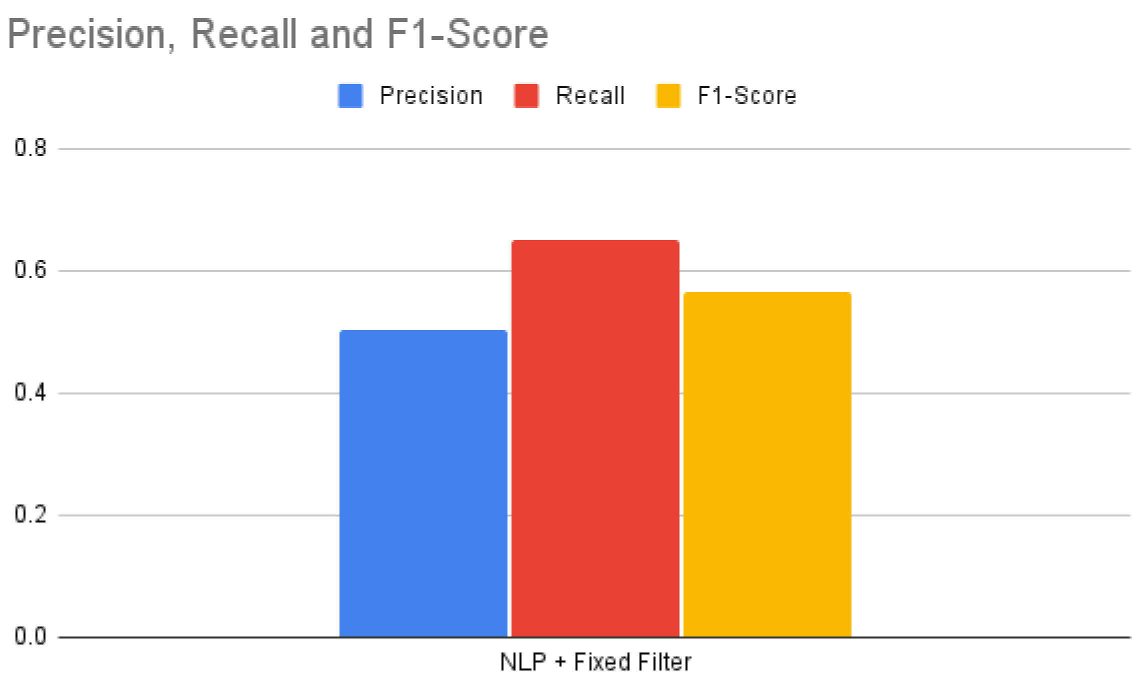


*exact numbers:

```
Model                          Precision    Recall    F1 Score
----------------------------  -----------  --------  ----------
urchade/gliner_base                0.7684    0.7684      0.7684
```

urchade/gliner_multi	0.7111	0.6531	0.6809
urchade/gliner_small-v1	0.7887	0.5957	0.6788
urchade/gliner_medium-v1	0.8052	0.7045	0.7515
urchade/gliner_large-v1	0.8272	0.7701	0.7976
urchade/gliner_small-v2	0.6533	0.6282	0.6405
urchade/gliner_medium-v2	0.6203	0.7424	0.6759
urchade/gliner_large-v2	0.5974	0.6571	0.6259
urchade/gliner_small-v2.1	0.8182	0.5934	0.6879
urchade/gliner_medium-v2.1	0.7931	0.7753	0.7841
urchade/gliner_large-v2.1	**0.8022**	**0.8295**	**0.8156**
urchade/gliner_multi-v2.1	0.8714	0.6703	0.7578

Fixed-Filter Score for First Dataset

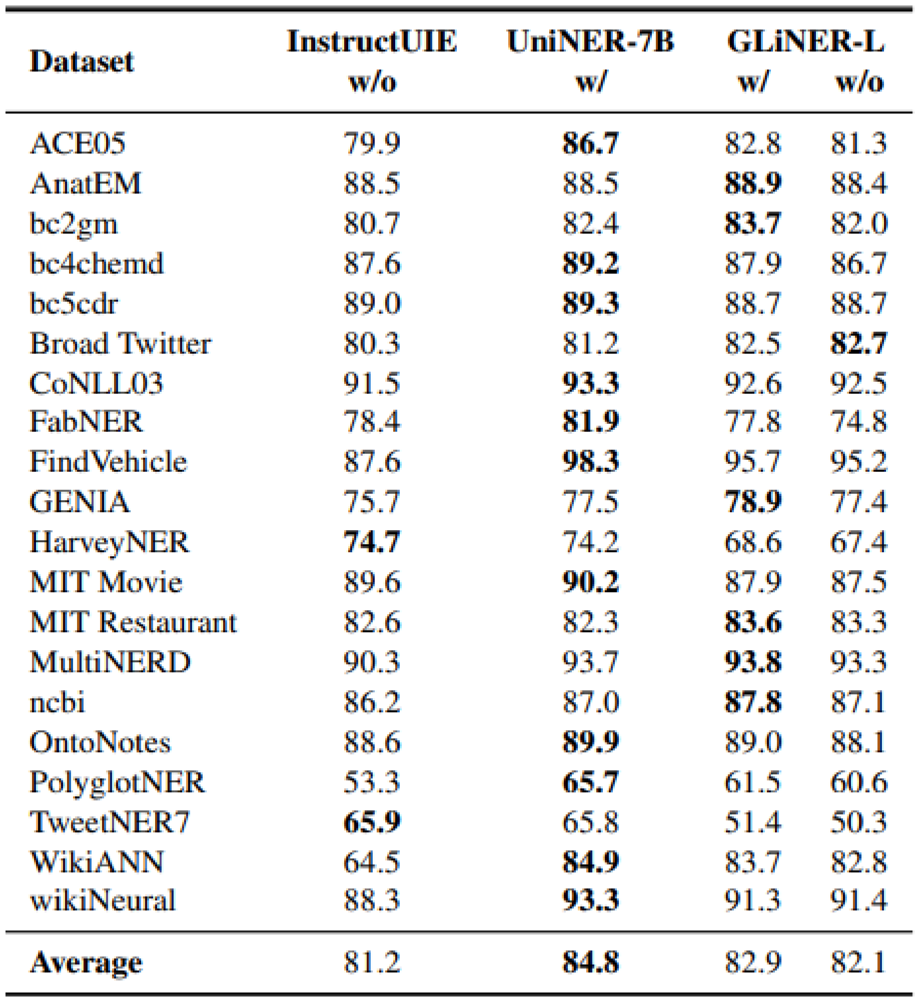


*exact numbers:

```
"precision": 0.5023255813953489
"recall": 0.6498194945848376,
"f1": 0.5666316894018887
```

GLiNER Paper Results

One thing that I want to compare the results from these tests are with the GLiNER paper. In one section of the paper that I will attach below they test GLiNER on datasets and we can get an idea of what F1 score was expected. The datasets used in the paper were also multilingual.

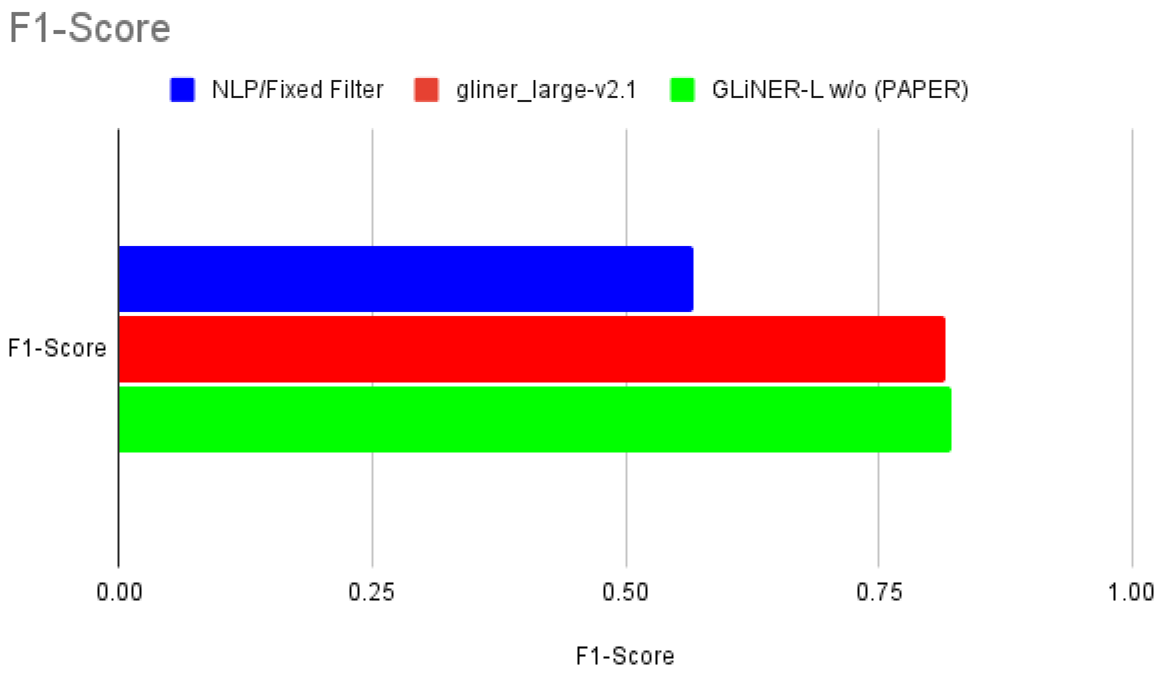


Dataset	**InstructUIE**	**UniNER-7B**	**GLiNER-L**	
	w/o	**w/**	**w/**	**w/o**
ACE05	79.9	**86.7**	82.8	81.3
AnatEM	88.5	88.5	**88.9**	88.4
bc2gm	80.7	82.4	**83.7**	82.0
bc4chemd	87.6	**89.2**	87.9	86.7
bc5cdr	89.0	**89.3**	88.7	88.7
Broad Twitter	80.3	81.2	82.5	**82.7**
CoNLL03	91.5	**93.3**	92.6	92.5
FabNER	78.4	**81.9**	77.8	74.8
FindVehicle	87.6	**98.3**	95.7	95.2
GENIA	75.7	77.5	**78.9**	77.4
HarveyNER	**74.7**	74.2	68.6	67.4
MIT Movie	89.6	**90.2**	87.9	87.5
MIT Restaurant	82.6	82.3	**83.6**	83.3
MultiNERD	90.3	93.7	**93.8**	93.3
ncbi	86.2	87.0	**87.8**	87.1
OntoNotes	88.6	**89.9**	89.0	88.1
PolyglotNER	53.3	**65.7**	61.5	60.6
TweetNER7	**65.9**	65.8	51.4	50.3
WikiANN	64.5	**84.9**	83.7	82.8
wikiNeural	88.3	**93.3**	91.3	91.4
Average	81.2	**84.8**	82.9	82.1

[Summary of the paper] The chart in Table 4 displays the performance of various models on NER datasets after in-domain supervised fine-tuning. It includes results for

GLiNER models with and without prior training on the Pile-NER dataset, alongside other models such as InstructUIE and UniNER-7B.

Quality Comparison of Techniques



After looking at the results we can see that clearly the AI-based filter worked much better in this case. Out of all the models, gliner_large-v2.1 is the most reliable at filtering PII. We can also see that it also performed very similarly to the paper. Before I come to a conclusion I also tested one more slightly different dataset.

Overview of the Second Dataset

To test the two filters again, I used another dataset that is quite different than the first one. A section of the second dataset is attached below:

```
[
    {
        "question": "Which of these circuits is located at a higher
latitude, Silverstone Circuit, Hockenheimring or Hungaroring?",
        "stripped_question": "Which of these circuits is located at a
higher latitude, <circuits.name>, <circuits.name> or <circuits.name>?",
     },
```

There are a few key things to note about the way this data is presented. This time we have a couple sentences as variables. The variable “question” has the text we want to take and filter out potential PII, or in this case things like names of circuits or drivers. Then I added another variable called “stripped_question” that has the same sentence as the variable before except it is filtered out with things like <circuits.name> instead of Silverstone Circuit for example.

This format lets me take “question” and run it into my program. After my program filters out what it thinks is PII then it can compare itself to “stripped_question” to see if it is filtered correctly and it can then provide a score. Also another important thing to note is that the programs are only going to “stripped_question” after it filters out the text to check if it's correct and unlike the first dataset, this one doesn’t give the labels it needs to remove.

AI-based Approach: GLiNER Models

I evaluated multiple versions of the GLiNER models again in my program to see which would be the best one to use. These models include the same ones as before:

- **- GLiNER Base**
- **- GLiNER Multi**
- **- GLiNER Small (v1, v2, v2.1)**
- **- GLiNER Medium (v1, v2, v2.1)**
- **- GLiNER Large (v1, v2, v2.1)**
- **- GLiNER Multi v2.1**

This program also had to be changed from taking labels directly from the datasets and putting it into the GLiNER model, to putting the labels I wanted to remove beforehand.

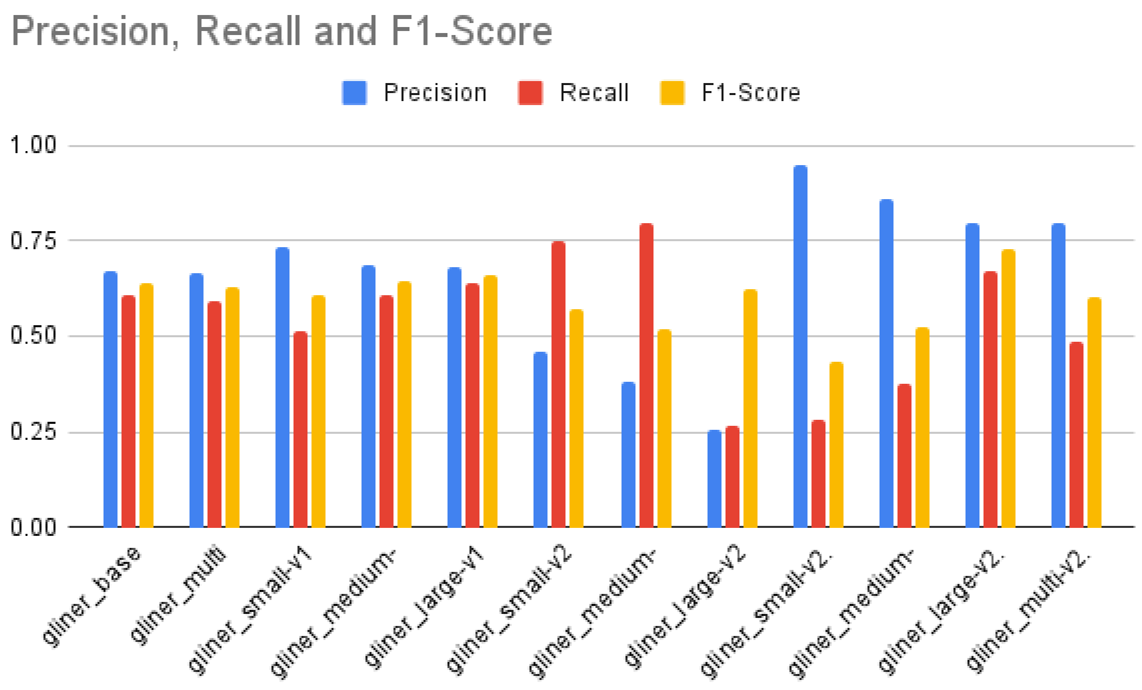
Rule-based Approach: Fixed Filter

Again the fixed filter used regular expressions and the NLP spacy to try and remove PII. This time the code had to be modified a bit to remove the types of PII that will be in this dataset. Most other things remained the same.

Results of the Second Dataset

Now that we went over some of the small changes made in both the AI-based approach using GLiNER models and the rule-based approach using spacy let's discuss the results. I am again going to be using the scores of precision, recall, and f1-score.

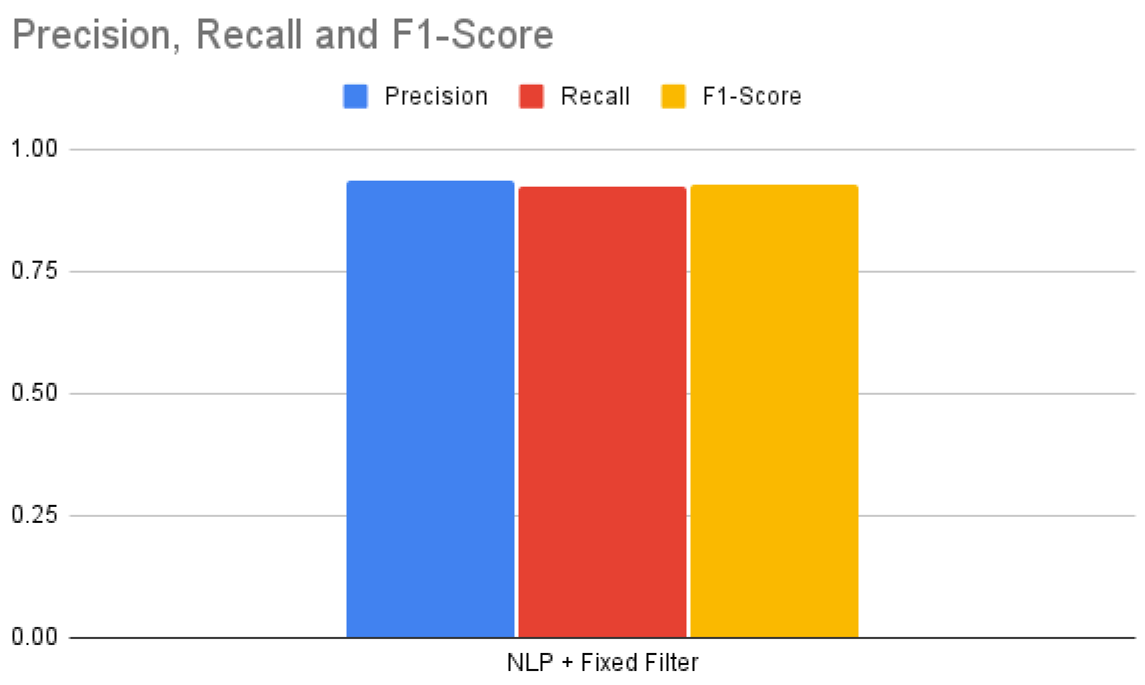
GLiNER Scores for the Second Dataset



*exact numbers:

```
Model                         Precision    Recall    F1 Score
---------------------------  -----------  --------  ----------
urchade/gliner_base               0.6724    0.6094      0.6393
urchade/gliner_multi              0.6667    0.5938      0.6281
urchade/gliner_small-v1           0.7333    0.5156      0.6055
urchade/gliner_medium-v1          0.6842    0.6094      0.6446
urchade/gliner_large-v1           0.6833    0.6406      0.6613
urchade/gliner_small-v2           0.4615    0.7500      0.5714
urchade/gliner_medium-v2          0.3835    0.7969      0.5178
urchade/gliner_large-v2           0.2576    0.2656      0.2615
urchade/gliner_small-v2.1         0.9474    0.2812      0.4337
urchade/gliner_medium-v2.1        0.8571    0.3750      0.5217
urchade/gliner_large-v2.1        0.7963    0.6719      0.7288
urchade/gliner_multi-v2.1         0.7949    0.4844      0.6019
```

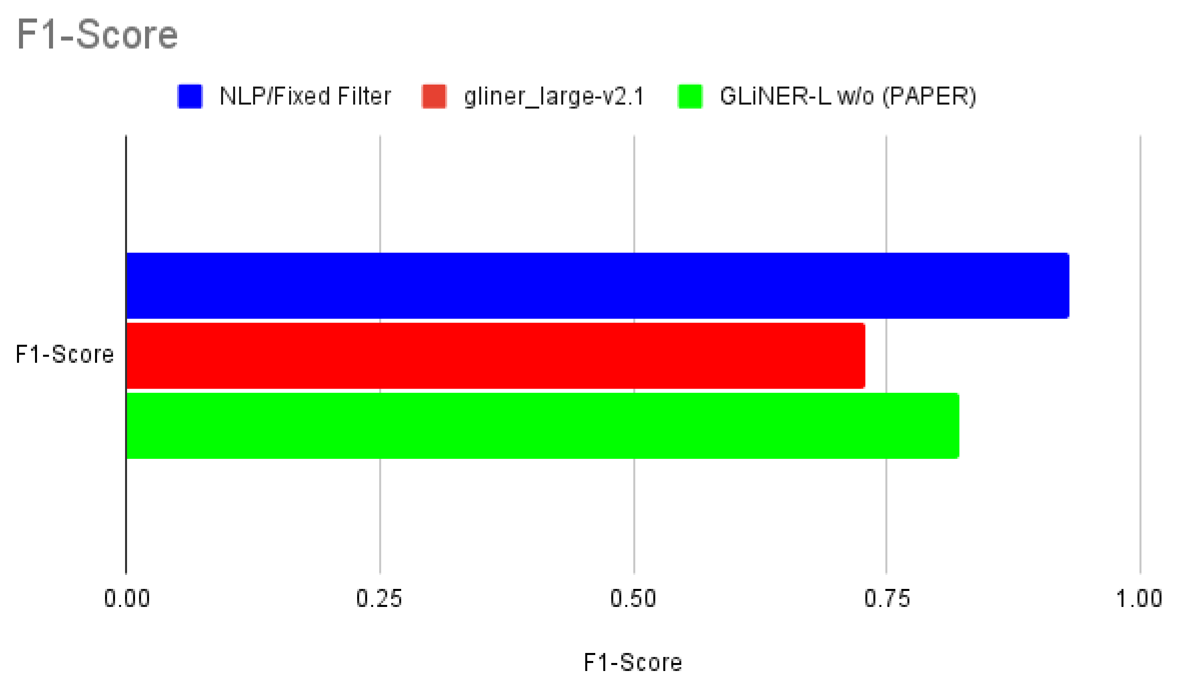
Fixed-Filter Score for Second Dataset



*exact numbers:

```
"precision": 0.9365079365079365,
  "recall": 0.921875,
  "f1": 0.9291338582677166,
```

Quality Comparison of Techniques



After looking at these new results we can see that a lot has changed from the first test. First we can see that this time the gliner_large-v2.1 is not performing as much as the original paper. It has gone down a little bit overall but not by too much. On the other hand the fixed filter's score has gone up significantly compared to last time. There are multiple reasons for all these changes that we will now discuss.

Conclusion

In this report, I evaluated the performance of AI-based and rule-based approaches for filtering Personally Identifiable Information (PII) from two distinct datasets. Each dataset presented unique challenges, highlighting the strengths and weaknesses of each approach under different conditions.

- **First Dataset: Multilingual and Label-Rich Data**

The first dataset consisted of tokenized text with explicit Named Entity Recognition (NER) labels, identifying various types of PII such as names, ID card numbers, and credit card numbers. The multilingual nature of the dataset added complexity, which the AI-based GLiNER models handled effectively. The key reason for GLiNER's superior performance on this dataset is its ability to extract the labels directly from the dataset. Since the GLiNER model can practically understand any label, it was able to accurately recognize and filter out diverse types of PII, leading to high precision, recall, and F1 scores.

In contrast, the rule-based approach using fixed filters struggled with the multilingual aspect and the variety of PII types. The fixed filters, which rely on predefined patterns and keyword matching, could not adapt as flexibly to the diverse labels in the dataset. As a result, the rule-based approach had a much lower precision and recall compared to the AI-based approach.

- **Second Dataset: Structured Questions and SQL Queries**

The second dataset presented a different challenge. It consisted of natural language questions and corresponding SQL queries, with the task being to identify and filter specific entities such as circuit names or driver names. Unlike the first dataset, this one did not provide direct labels for each entity, making it harder for the AI-based GLiNER models to perform well. The lack of direct label feeding resulted in a slight drop in the AI models' performance, with lower precision, recall, and F1 scores compared to the first dataset.

On the other hand, the rule-based approach excelled in this scenario. The fixed filters, with their ability to use regular expressions and keyword matching, were well-suited for identifying specific patterns in the structured questions. As a result, the rule-based approach achieved a significantly higher F1 score, demonstrating its effectiveness when the types of PII are well-defined and follow predictable patterns.

Strengths and Weaknesses

The results highlight the strengths and weaknesses of both approaches:

- **AI-based Approach (GLiNER Models)**: This approach excels in scenarios where the dataset includes its own labels(although not needed) and a variety of PII types. Its flexibility and adaptability make it great for multilingual datasets and situations where PII can take many forms. On top of that the threshold variable can be adjusted to any number based on whatever a user wants. If they are more focused on recall and that no sensitive information is unfiltered, then they can lower the threshold. And if they care more about precision and accuracy then vice versa.
- **Rule-based Approach (Fixed Filters)**: This approach is highly effective when the PII types are well-defined and follow consistent patterns. It performed exceptionally well on the second dataset, where entities could be identified using regular expressions and keyword matching. However, it struggles with multilingual datasets and diverse PII types that do not follow fixed patterns.

Final Thoughts

In conclusion, the choice between AI-based and rule-based PII filtering depends on the type of the dataset. For multilingual and label-rich datasets, AI-based models like GLiNER can perform much better because of their flexibility and ability to handle diverse PII types. In contrast, for datasets with well-defined and PII patterns that you already know, rule-based approaches using fixed filters and NLPs can achieve higher accuracy and are more efficient.

By understanding the strengths and limitations of each approach, people can choose which method to employ for effective PII filtering in their specific contexts.

References

Zaratiana, U., Tomeh, N., Holat, P., & Thierry Charnois, T. (2023). *GLiNER: Generalist Model for Named Entity Recognition using Bidirectional Transformer*. arXiv. https://arxiv.org/pdf/2311.08526

Meulen, L. van der. (2021, August 12). *Remove personal information from a text with python‑part II‑Ner*. Medium. https://towardsdatascience.com/remove-personal-information-from-a-text-with-python-part-ii-ner-2e6529d409a6

https://huggingface.co/urchade/gliner_small-v2.1

https://www.kaggle.com/competitions/pii-detection-removal-from-educational-data/overview