PII Filter Implementation

Comparing AI-based and Rule-based Approaches

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"]]
}, …
```


The main things to note about this dataset is the ther is tokenized text with Named Entity Recognition (NER) labels provided. It is also a multilingual dataset.

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 NER models limited to predefined entities, GLiNER can identify any type of entity, making it adaptable for various tasks. It is also more efficient than large language models like GPT-3, using less computational resources.

Another feature in GLiNER is called threshold. 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
```

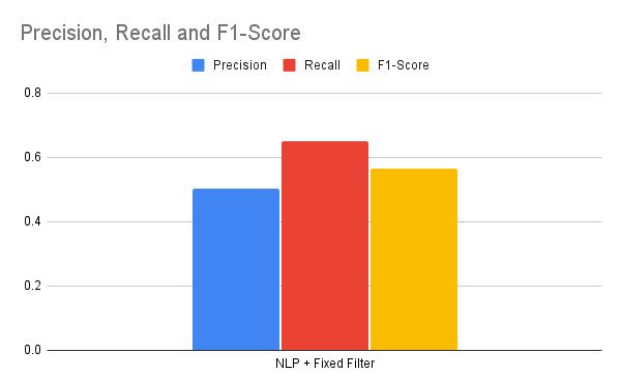
- The code also uses regular expressions for things like dates, times, email addresses, URLs, and postal codes. This works because these types of entities in text will almost always follow a pattern. Here is an example of one of the methods that used regular expressions to filter out postal codes. It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

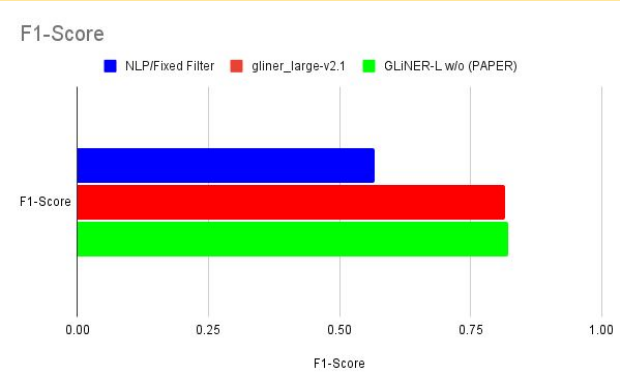
```
def remove_postal_codes(self, text):
        return re.sub(r'\b\d{5}(-\d{4})?\b', '<FILTERED>', text)
```

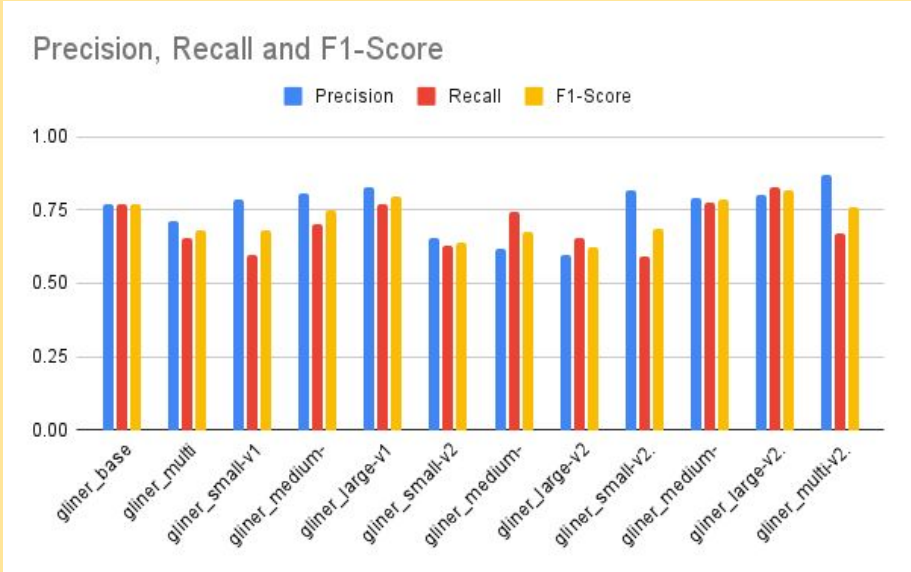
Metrics Used

- **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.
- **F1-score** provides a balance between precision and recall, giving a single metric for comparison.

Results of the First Dataset







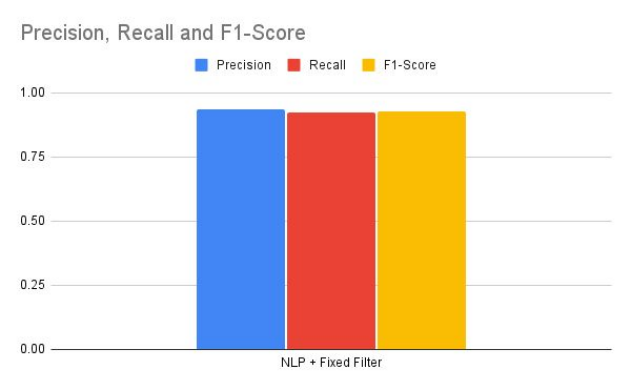
Overview of the Second Dataset

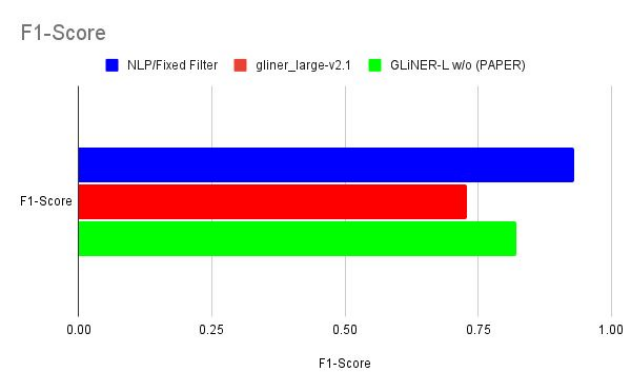
To test the two filters again, I used another dataset that is quite different than the first one. A section of the second dataset is attached below:

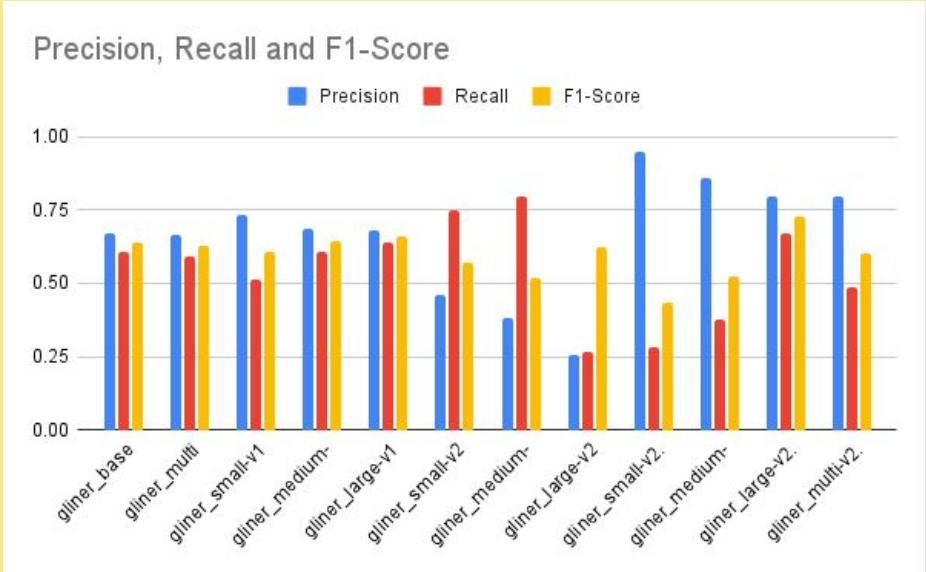

```
[
    {
        "question": "Which of these circuits is located at a higher latitude, Silverstone Circuit, Hockenheimring or Hungaroring?",
        "stripped_question": "Which of these circuits is located at a higher latitude, <circuits.name>, <circuits.name> or <circuits.name>?",
     },
```


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.

Results of the Second Dataset







Conclusion

The AI-based GLiNER models excelled in the first dataset, handling its multilingual and label-rich nature effectively, while the rule-based approach struggled. In contrast, with the second dataset, the rule-based method performed better due to its ability to handle structured questions and predictable patterns, where the GLiNER models experienced a slight performance drop without direct labels.

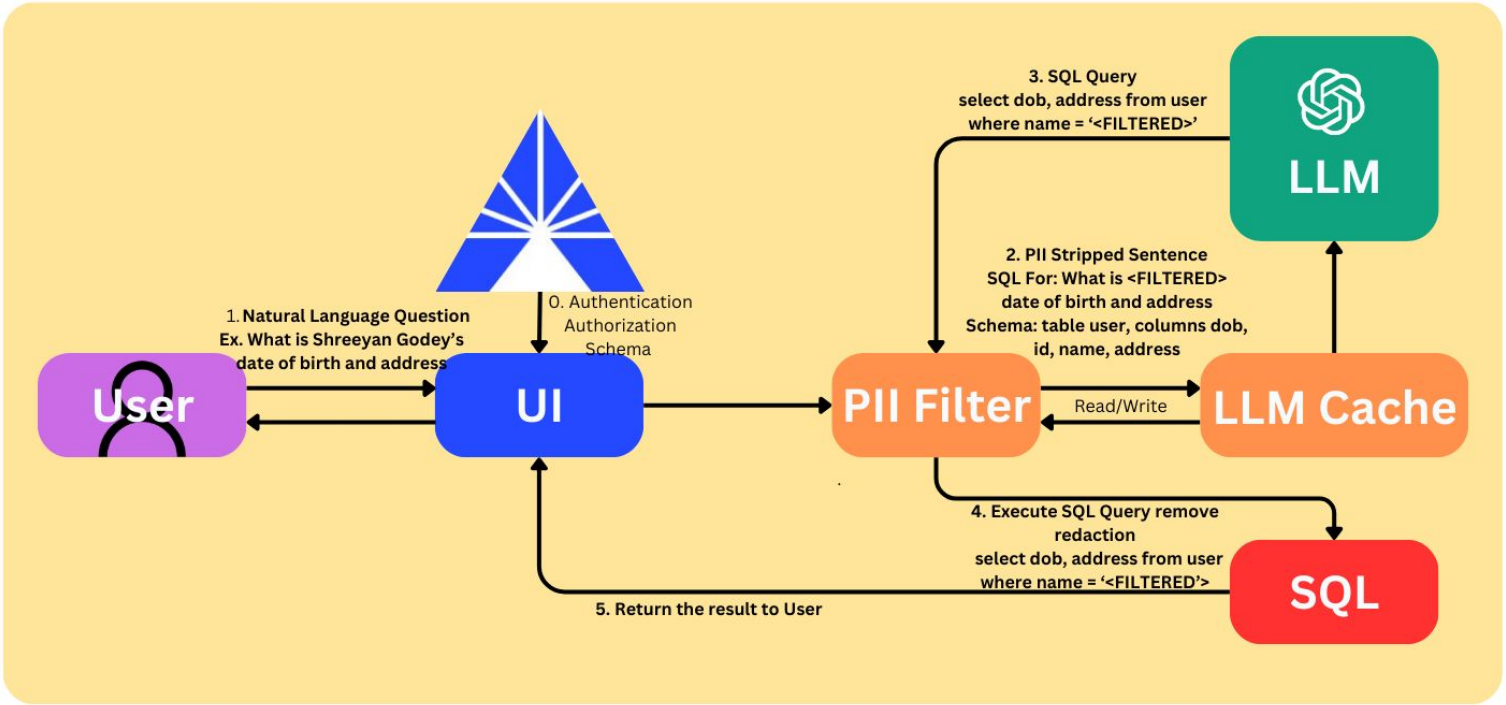
In conclusion, the choice between AI-based and rule-based PII filtering depends on the type of the dataset. For multilingual and label-rich datasets, AI-based models like GLiNER can perform much better because of their flexibility and ability to handle diverse PII types. In contrast, for datasets with 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.

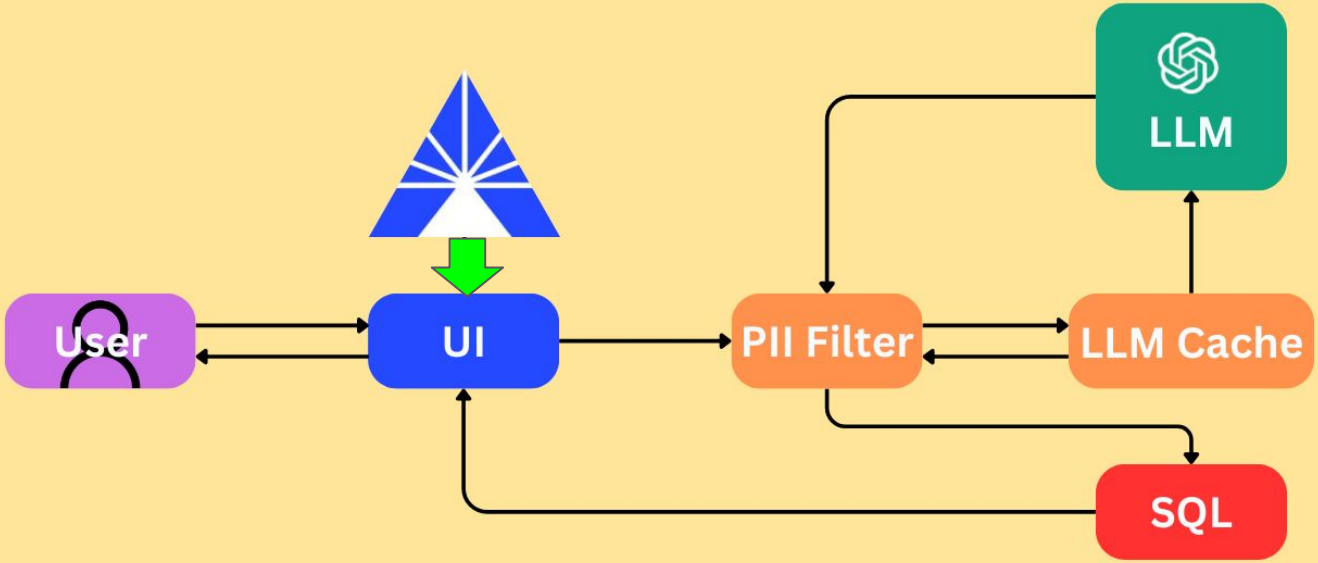
By understanding the strengths and limitations of each approach, people can choose which method to employ for effective PII filtering in their specific contexts.

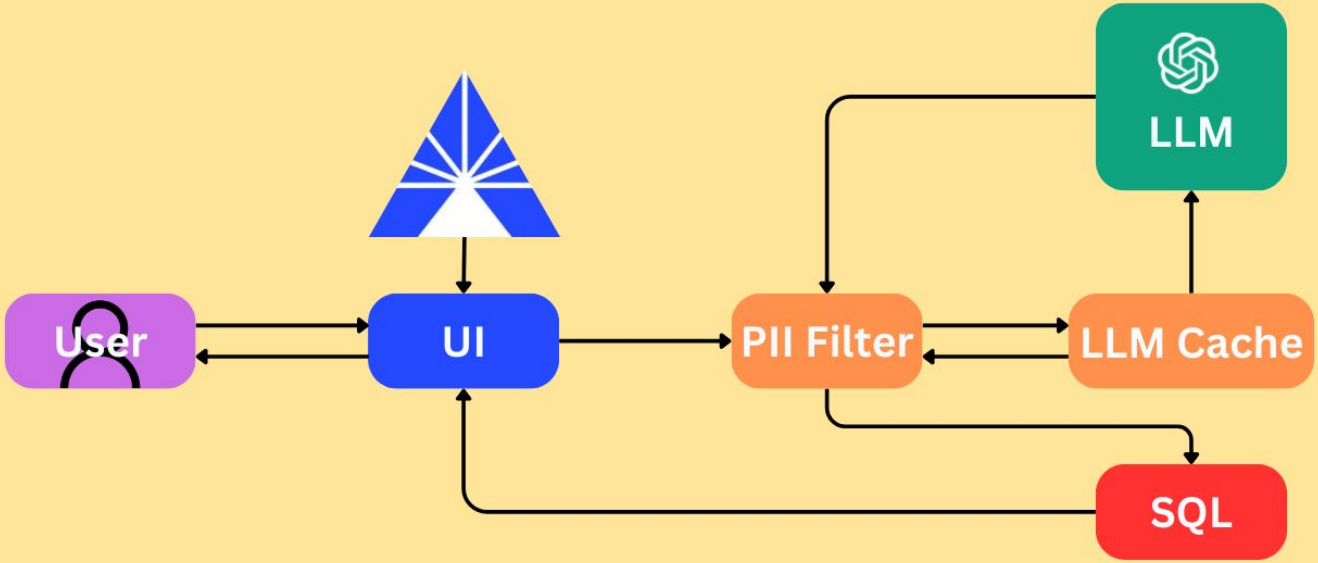
Strengths and Weaknesses

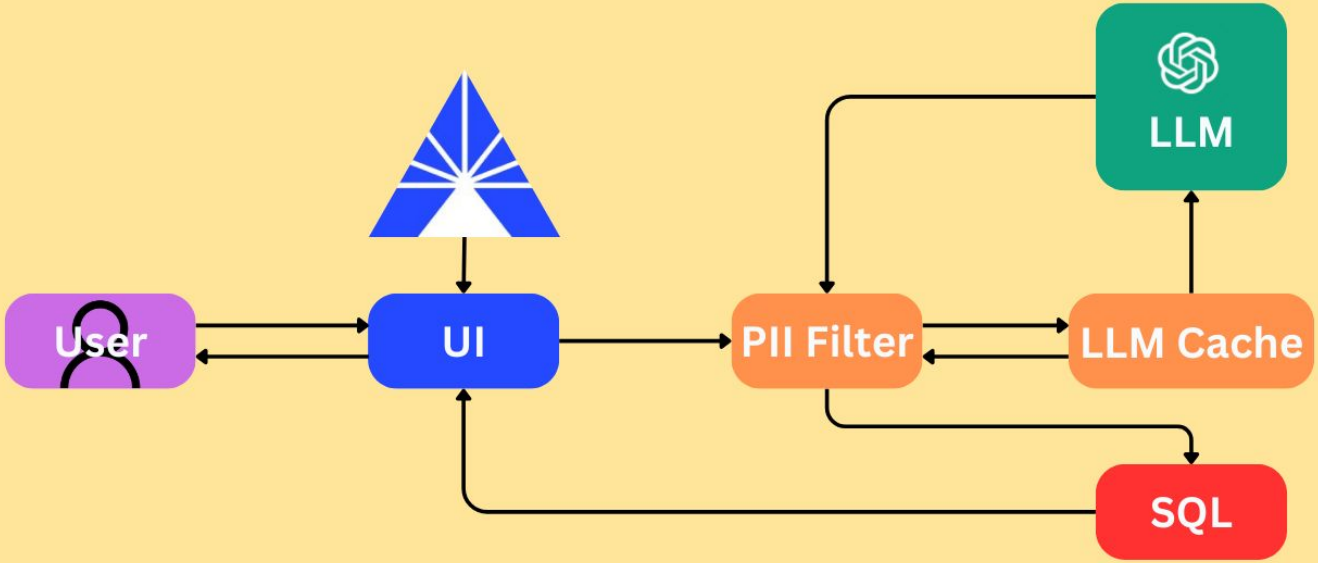
The results highlight the strengths and weaknesses of both approaches:

- **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.


3. SQL Query
select dob, address from user
where name = '<FILTERED>'
LLM
2. PII Stripped Sentence
SQL For: What is <FILTERED>
date of birth and address
Schema: table user, columns dob,
id, name, address
0. Authentication
Authorization
Schema
1. Natural Language Question
Ex. What is Shreeyan Godey's
date of birth and address
User
UI
PII Filter
Read/Write
LLM Cache
4. Execute SQL Query remove
redaction
select dob, address from user
where name = '<FILTERED'>
SQL
5. Return the result to User


LLM
User
UI
PII Filter
LLM Cache
SQL


LLM
User
UI
PII Filter
LLM Cache
SQL


LLM
User
UI
PII Filter
LLM Cache
SQL