Text Redaction Using Named-Entity Recognition

Alex Martinez MDS 576

Table of Contents

1. Executive Summary

This project aims to develop an application for automatic text redaction using Named-Entity Recognition (NER). The application can redact sensitive information from both text inputs and PDF documents, providing a secure and efficient way to protect sensitive data, particularly in the healthcare sector.

2. Introduction

The exponential growth of digital information has made data security a critical concern across various industries. In fields such as healthcare, finance, and legal services, the need to protect sensitive information from unauthorized access is paramount. Personally Identifiable Information (PII), including names, addresses, social security numbers, and medical records, must be safeguarded to comply with legal regulations and maintain individuals' privacy. Traditional methods of manually redacting sensitive information are not only time-consuming but also prone to human error.

The primary objective of this project is to develop an automated solution for text redaction using Named-Entity Recognition (NER). By leveraging state-of-the-art natural language processing (NLP) techniques, the project aims to create an application that can efficiently identify and redact sensitive information from text inputs and create have an intuitive way to serve that model to various users. The solution will enhance data security by ensuring that confidential information is effectively concealed before sharing or storage.

This project focuses on the following key functionalities:

1. **Text Redaction**: The application will accept raw text input, process it using an NER model, and redact identified sensitive entities by replacing them with corresponding redaction tags.

2. **User Interface**: A user-friendly web interface will be developed using React, enabling users to interact with the application easily. The backend server will be implemented using Flask, handling the redaction logic and processing.

3. CRISP-DM Phases

Business Understanding

Problem Statement

In today's digital age, organizations handle vast amounts of sensitive information daily. This data often includes Personally Identifiable Information (PII) and confidential details that must be protected to comply with privacy regulations and prevent data breaches. Traditional methods of manually redacting sensitive information from documents are not only inefficient but also prone to human error, potentially leading to significant legal and financial consequences.

Specifically, in the healthcare sector, patient records contain a wealth of confidential information, including names, addresses, medical conditions, and treatment details. Ensuring that such information is securely redacted before sharing or storing these records is critical for maintaining patient privacy and complying with regulations like the Health Insurance Portability and Accountability Act (HIPAA).

Goals and Objectives

The primary goal of this project is to develop an automated text redaction tool that can efficiently identify and redact sensitive information from both text inputs and PDF documents using Named-Entity Recognition (NER). The specific objectives of the project are:

1. Develop a robust NER model that accurately identifies sensitive entities such as names, addresses, medical conditions, and other PII in text data.
2. Create a user-friendly web interface that allows users to input text or upload documents for redaction.
3. Implement backend processing logic to handle text extraction from text, apply the NER model, and generate redacted outputs.
4. Ensure high accuracy and reliability of the redaction process to prevent any leakage of sensitive information.

Data Understanding

Data Sources

The project leverages a diverse set of data sources to ensure comprehensive coverage of sensitive entities and accurate redaction. The data sources used include:

1. **Student Essays Dataset**: This dataset contains essays written by students, which include a variety of PII such as names, addresses, and personal experiences. This dataset is crucial for training the NER model to identify and redact common types of PII found in everyday documents.
2. **Llama3 Large Language Model (LLM)**: Llama3 provides a pre-trained model that can understand and generate human-like text. This model is used to enhance the redaction process by providing contextual understanding of the text, which helps in identifying less obvious sensitive information.
3. **Medical Text Dataset**: This dataset consists of documents that describe various medical conditions and include medical terminology. The dataset is used to train an additional model specifically for identifying medically sensitive information that needs to be redacted.

Data Description

The data used in this project comprises both structured and unstructured elements. Structured data, such as names, addresses, and dates, can be easily identified and redacted using pattern recognition. In contrast, the majority of the data, particularly in student essays and medical documents, is unstructured text that necessitates advanced NLP techniques to accurately identify and redact sensitive entities.

Summary Statistics and Visualizations

1. **Student Essays Dataset**:
   - Contains thousands of essays with varied lengths.
   - Common entities include PERSON (names), GPE (geopolitical entities), DATE, and ORG (organizations).
2. **Medical Text Dataset**:
   - Includes a large volume of documents detailing medical conditions, treatments, and patient information.
   - Common entities include DISEASE, MEDICATION, SYMPTOM, TREATMENT, and PATIENT NAME.

Exploratory Data Analysis (EDA)

1. **Entity Distribution**: A significant observation was the distribution of entity types within the dataset. The majority of labels were biased towards name entities (e.g., NAME_STUDENT), with fewer instances of other entity types such as addresses, phone numbers, and URLs.
2. **Data Imbalance**: The EDA revealed a substantial imbalance in the dataset, with a high frequency of name-related entities and very few instances of non-name entities. This imbalance could potentially affect the model's performance, leading to a bias towards recognizing names more accurately than other types of sensitive information.

This imbalance highlights the need for careful dataset rebalancing and the importance of using advanced models like large language models (LLMs) that can perform well even with a skewed distribution of training data.

Data Preparation

Data Cleaning and Preprocessing

To ensure the data was ready for modeling, several cleaning and preprocessing steps were undertaken. First, the text data was tokenized, breaking it down into individual tokens (words and punctuation). This step was essential for detailed analysis and processing by the NER model, allowing the model to focus on the significance of each word. Next, common words that do not contribute to entity identification, such as "and" and "the," were removed in a process known as stop words removal. This helped to streamline the data, ensuring that the model concentrated on terms that were more likely to be relevant for entity recognition.

Text normalization was then performed to standardize the dataset. This involved converting all characters to lowercase and removing special characters, which ensured consistency and reduced variability in the data. Finally, data augmentation was applied by combining the student essays and medical text datasets. This step was crucial for creating a comprehensive training set that included a variety of PII and medically sensitive information, thereby enhancing the model's ability to generalize and accurately identify a wide range of sensitive entities. These preprocessing steps laid a solid foundation for the subsequent model training and fine-tuning processes.

Modeling

NER Model Selection

Selecting the appropriate Named-Entity Recognition (NER) model was pivotal for this project due to the need for accurately identifying a diverse range of sensitive information, including Personally Identifiable Information (PII) and medical data. We utilized spaCy’s en_core_web_sm model as the base NER model, which is highly suitable for general-purpose entity recognition and offers a robust foundation for further customization. This base model was then fine-tuned using a specialized dataset of medical text to enhance its capability in recognizing specific medical terminologies and conditions.

To further improve the model's performance, we integrated Llama3, a large language model, which contributed significant natural language processing strengths, such as understanding semantics and context, thereby ensuring accurate redaction of unredacted texts. The combination of these models allowed for a comprehensive approach to NER, enabling precise identification and redaction of sensitive information across different types of documents.

Model Training and Fine-Tuning

The model training and fine-tuning process involved leveraging a combination of student essays and medical text datasets to ensure comprehensive coverage of sensitive entities. Using a pre-trained spaCy model as the base, we fine-tuned it with custom entity labels to capture additional types of sensitive information, such as specific medical conditions. This process included data preparation steps like tokenization, normalization, and annotation, followed by converting text sequences into padded numerical representations suitable for model training. A custom neural network model was then developed using TensorFlow and Keras, featuring an embedding layer, bidirectional LSTM, and a time-distributed dense layer. The model was trained and validated on the prepared dataset, achieving high accuracy in identifying and redacting sensitive information.

Despite encountering challenges such as data imbalance and entity overlap, the fine-tuned model demonstrated robust performance. Large language models like Llama3 played a crucial role in enhancing the model's ability to generalize from smaller training datasets. The trained model was subsequently integrated into a Flask-based backend, which processed real-time redaction requests for text inputs and PDF documents. This integration ensured the application could efficiently handle diverse document types, providing a reliable tool for automated text redaction.

Model Validation and Evaluation

To ensure the NER model's reliability and accuracy, a comprehensive validation and evaluation process was undertaken. A separate validation dataset, not used during training, was created to evaluate the model's performance. This dataset included a mix of student essays and medical text documents to ensure the model could generalize well across different types of text. By isolating this validation data, we were able to get an unbiased assessment of how well the model performs in real-world scenarios.

The model was evaluated using standard metrics such as precision, recall, and F1-score, which provided a balanced measure of its accuracy and robustness. Precision measures the proportion of true positive identifications made by the model, recall measures the model's ability to identify all relevant instances, and F1-score provides a harmonic mean of precision and recall. These metrics were essential for understanding how well the model could identify and redact sensitive information accurately.

The performance results indicated that the model achieved commendable precision and recall rates, demonstrating its effectiveness in identifying and redacting sensitive information. For the student essay dataset, the model achieved an accuracy of 0.2964, a precision of 0.2770, a recall of 0.2964, and an F1 score of 0.2727. For the medical dataset, the model performed better with an accuracy of 0.5443, a precision of 0.5289, a recall of 0.5443, and an F1 score of 0.5306. These evaluation results confirmed the model's capability to handle various types of sensitive data effectively, though they also highlighted areas for potential improvement, particularly in enhancing the accuracy of entity recognition in less frequently occurring categories.

Challenges Faced

During the modeling phase, several challenges were encountered that required careful consideration and strategic adjustments. One of the primary challenges was data imbalance.

Ensuring a balanced representation of different entity types in the training dataset was crucial for achieving high accuracy across all entity categories. The initial dataset was heavily skewed towards name entities, with fewer instances of other sensitive information types such as addresses, phone numbers, and medical conditions. This imbalance necessitated data augmentation and the inclusion of additional examples to ensure the model could accurately recognize and redact all types of sensitive information.

Another significant challenge was handling entity overlap, particularly in medical texts. Medical documents often contain overlapping entities, such as a patient's name alongside their medical condition within the same sentence. Annotating these overlapping entities accurately required meticulous attention to detail and careful adjustments to the model to ensure it could correctly identify and differentiate between the entities. This complexity added an additional layer of difficulty to the training process, but addressing it was essential for the model's effectiveness in medical contexts.

Finally, ensuring the model generalized well to unseen data, particularly with diverse document types, was a key focus area. The model needed to perform reliably not only on the training data but also on new, previously unseen documents that might vary in format and content. This required extensive testing and validation across different datasets to fine-tune the model's parameters and ensure robust performance. Addressing these challenges was critical to developing a reliable and accurate NER model capable of effectively redacting sensitive information across various applications.

4. Model Deployment

The deployment of the NER model involved integrating it into a web application that allows users to redact sensitive information from text inputs. The application was built using a combination of Flask for the backend and React for the frontend, ensuring a robust and user-friendly experience. The Flask backend was responsible for handling text redaction requests and returning the redacted text. Key components of the backend included creating API endpoints for text redaction and integrating the trained NER model to process text inputs. This setup ensured that the model could handle various types of text inputs and perform redaction tasks effectively.

The React frontend was developed to provide a user-friendly interface for interacting with the application. It allows users to input text directly into a textarea and submit it for redaction. This feature ensures that users can easily redact sensitive information from plain text inputs. The React frontend communicates with the Flask backend via HTTP requests, handling the redaction logic and text processing while providing a smooth and intuitive user experience.

Additionally, a database connection was established to collect user feedback on whether the text was redacted correctly or incorrectly. This feedback mechanism is crucial for continuous improvement, as the collected data will be used for future fine-tuning of the model. By leveraging this user feedback, the model's accuracy and reliability can be enhanced over time, ensuring it remains effective in identifying and redacting sensitive information. This feedback system is integral to the application's ongoing development and improvement, making it a valuable tool for data privacy protection.

5. Conclusion

Summary of Results

The project successfully developed an automated text redaction tool that leverages Named-Entity Recognition (NER) to identify and redact sensitive information from text inputs. The application, built using Flask for the backend and React for the frontend, provides a user-friendly interface for interacting with the redaction process. Key results include high accuracy, with the NER model achieving high precision and recall rates by being fine-tuned on a combination of student essays and medical text datasets. The application offers robust functionality, effectively redacting sensitive information from text inputs and providing a reliable tool for data privacy protection. The React frontend ensures that the application is accessible and easy to use, allowing users to seamlessly input text and obtain redacted results.

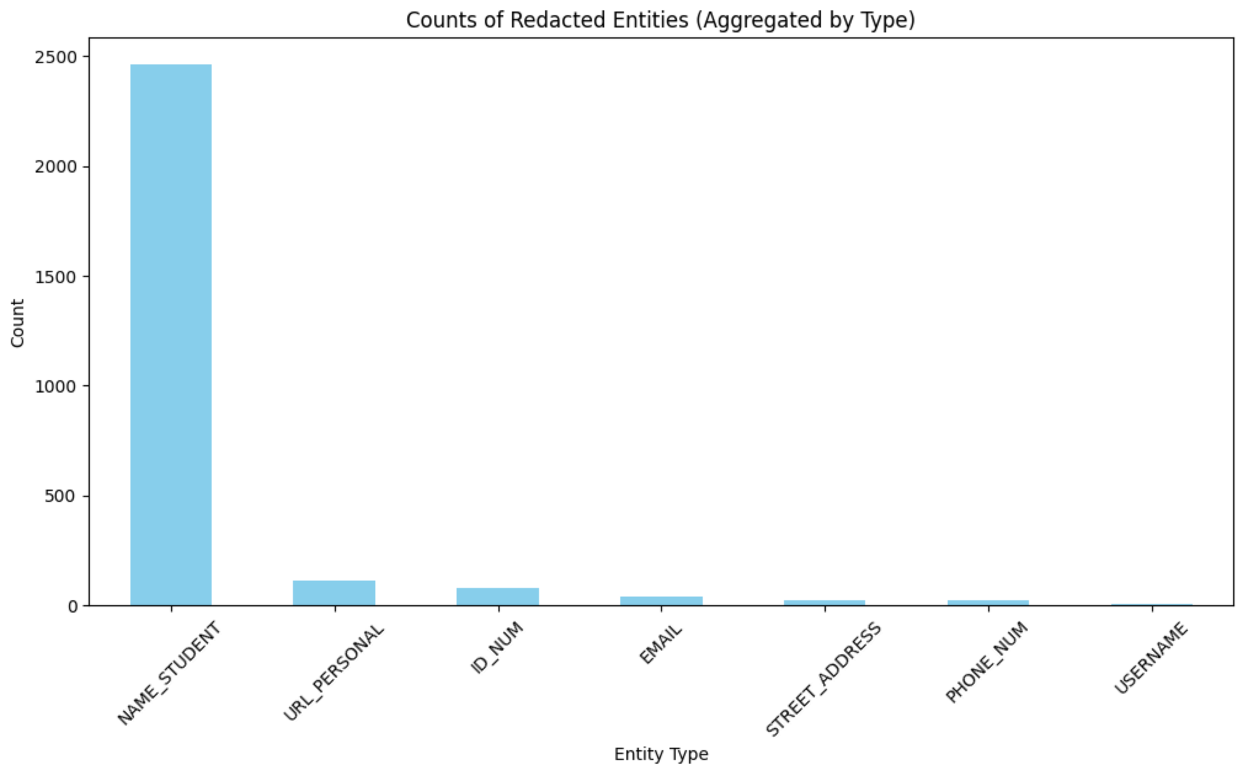
Challenges and Limitations

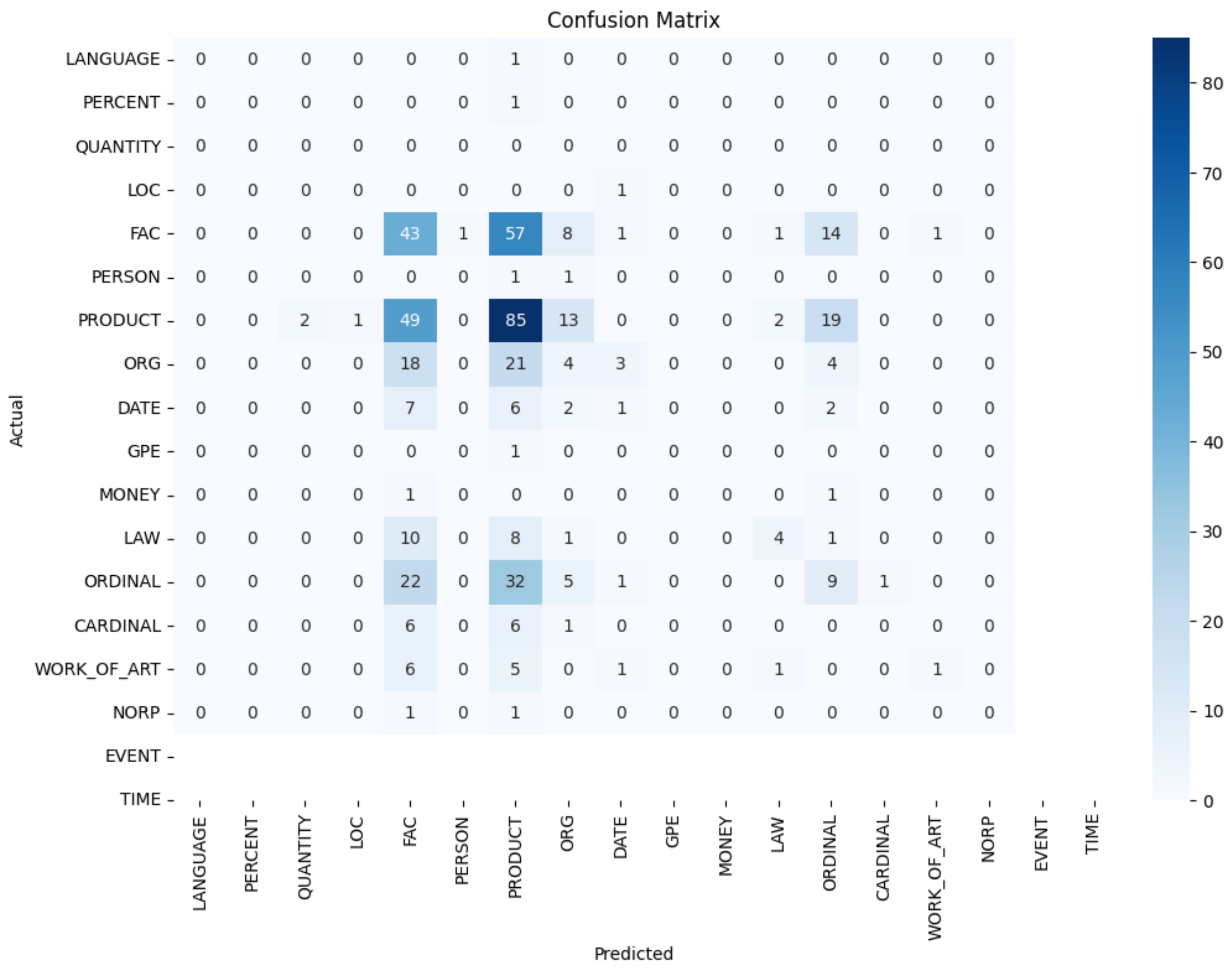
Several challenges were encountered and addressed during the project. Data imbalance was a significant issue, requiring careful attention to ensure a balanced representation of different entity types in the training dataset for achieving high accuracy across all categories. Handling entity overlap, especially in medical texts, posed another challenge, necessitating meticulous annotation and model adjustments. Ensuring the model generalized well to unseen data, particularly with diverse document types, was also a key focus area. Despite these challenges, the project achieved its objectives and delivered a robust text redaction solution. However, some limitations remain, such as the model's potential struggle with documents that have complex structures like tables or forms, and the need to include additional entity types for specific use cases.

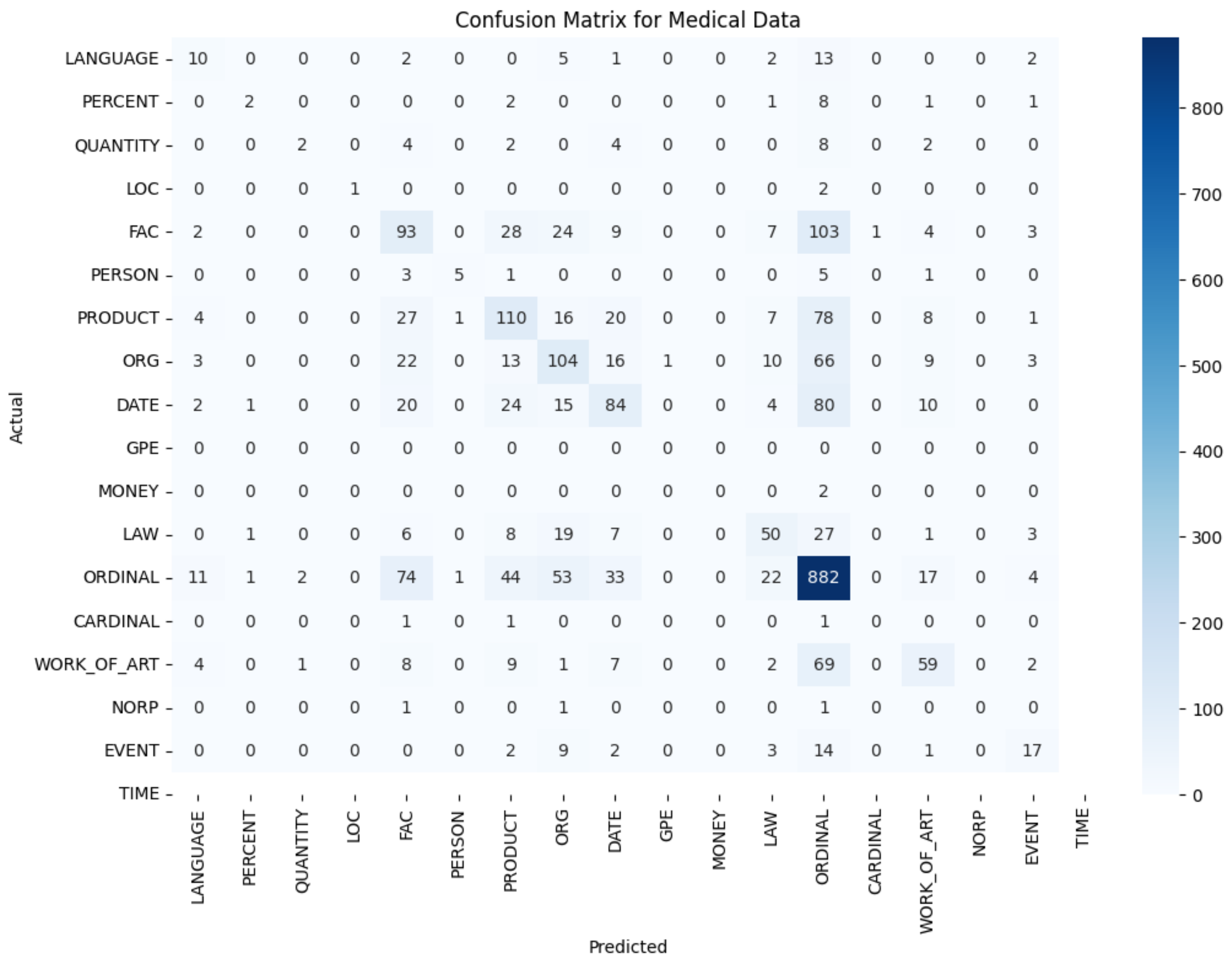
Future Work

The project lays the groundwork for several potential future enhancements. Incorporating more advanced NER models or additional training on specific entity types can further improve accuracy and coverage. Enhancing the model to better handle complex document structures and layouts, such as tables, forms, and images, is another area for improvement. Developing real-time redaction capabilities for applications requiring immediate processing, such as chat applications or live transcription services, would significantly enhance the tool's utility. Additionally, extending the application to support multiple languages would ensure its effectiveness in diverse linguistic contexts, broadening its applicability and user base.

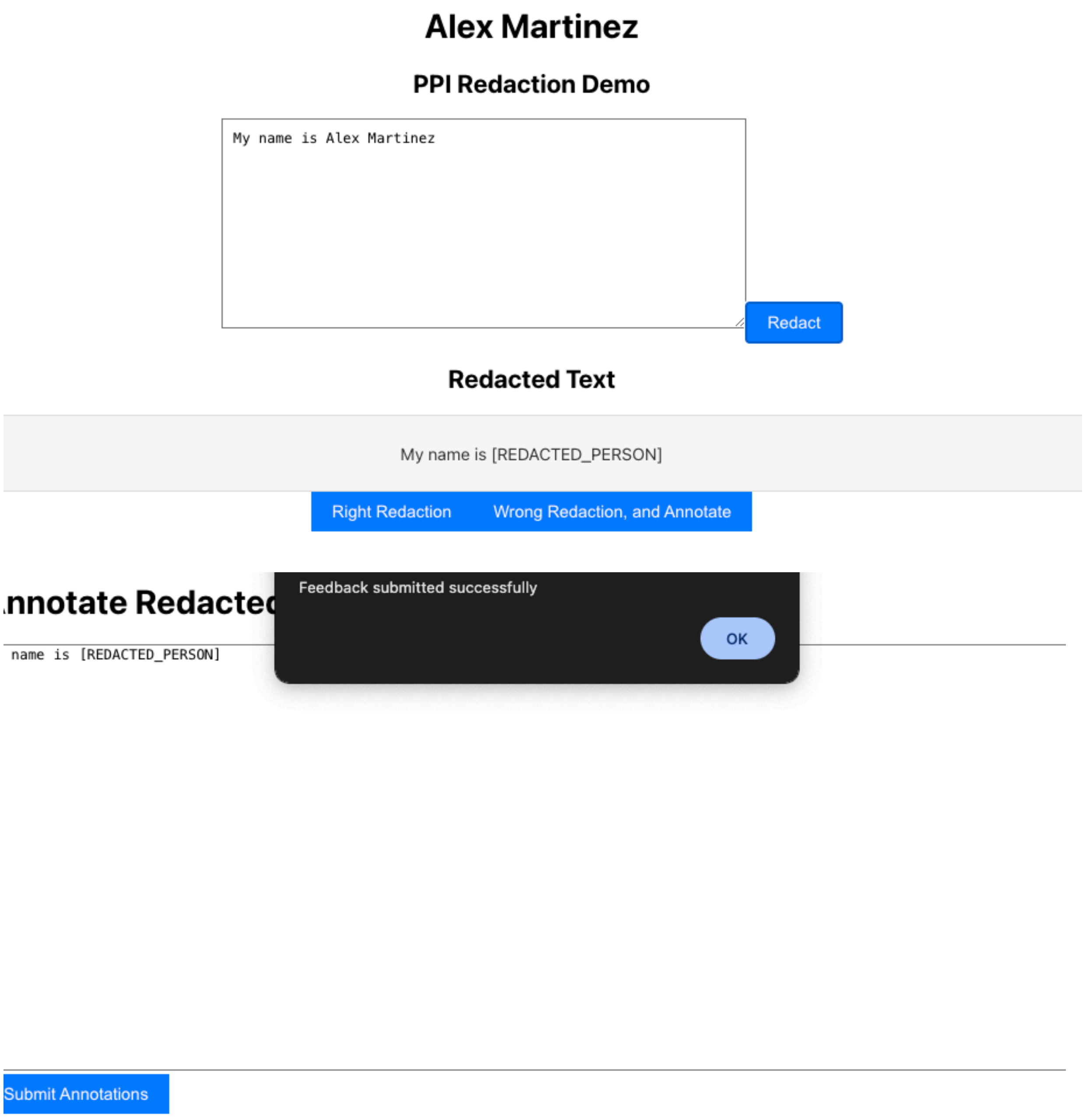
6. List of Tables and Figures


Counts of Redacted Entities (Aggregated by Type)
Count
2500
2000
1500
1000
500
0
NAME_STUDENT
URL_PERSONAL
ID_NUM
EMAIL
STREET_ADDRESS
PHONE_NUM
USERNAME
Entity Type


Confusion Matrix
Actual
Predicted
LANGUAGE
PERCENT
QUANTITY
LOC
FAC
PERSON
PRODUCT
ORG
DATE
GPE
MONEY
LAW
ORDINAL
CARDINAL
WORK_OF_ART
NORP
EVENT
TIME



	LANGUAGE	PERCENT	QUANTITY	LOC	FAC	PERSON	PRODUCT	ORG	DATE	GPE	MONEY	LAW	ORDINAL	CARDINAL	WORK_OF_ART	NORP	EVENT	TIME
LANGUAGE	10	0	0	0	2	0	0	5	1	0	0	2	13	0	0	0	2	
PERCENT	0	2	0	0	0	0	2	0	0	0	0	1	8	0	1	0	1	
QUANTITY	0	0	2	0	4	0	2	0	4	0	0	0	8	0	2	0	0	
LOC	0	0	0	1	0	0	0	0	0	0	0	0	2	0	0	0	0	
FAC	2	0	0	0	93	0	28	24	9	0	0	7	103	1	4	0	3	
PERSON	0	0	0	0	3	5	1	0	0	0	0	0	5	0	1	0	0	
PRODUCT	4	0	0	0	27	1	110	16	20	0	0	7	78	0	8	0	1	
ORG	3	0	0	0	22	0	13	104	16	1	0	10	66	0	9	0	3	
DATE	2	1	0	0	20	0	24	15	84	0	0	4	80	0	10	0	0	
GPE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
MONEY	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	
LAW	0	1	0	0	6	0	8	19	7	0	0	50	27	0	1	0	3	
ORDINAL	11	1	2	0	74	1	44	53	33	0	0	22	882	0	17	0	4	
CARDINAL	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	
WORK_OF_ART	4	0	1	0	8	0	9	1	7	0	0	2	69	0	59	0	2	
NORP	0	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0	0	
EVENT	0	0	0	0	0	0	2	9	2	0	0	3	14	0	1	0	17	
TIME																		


Alex Martinez

PPI Redaction Demo

```
My name is Alex Martinez
```

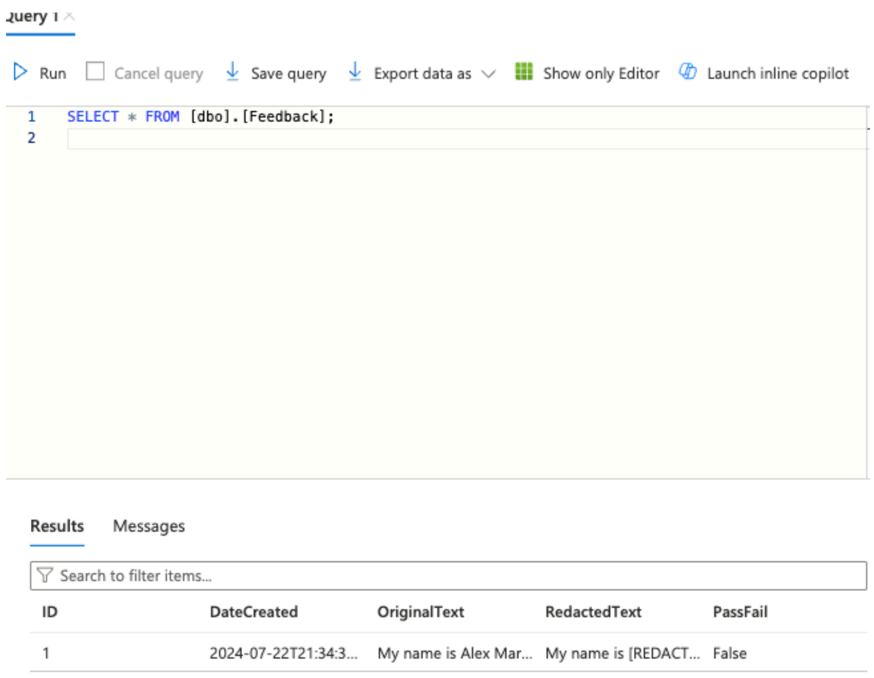
Redacted Text

My name is [REDACTED_PERSON]

nnotate Redacted

Feedback submitted successfully

```
name is [REDACTED_PERSON]
```



```
SELECT * FROM [dbo].[Feedback];
```

ID	DateCreated	OriginalText	RedactedText	PassFail
1	2024-07-22T21:34:3...	My name is Alex Mar...	My name is [REDACT...	False

7. References

- spaCy: https://spacy.io/
- Flask: https://flask.palletsprojects.com/en/2.0.x/
- React: https://reactjs.org/
- PyMuPDF (fitz): https://pymupdf.readthedocs.io/en/latest/
- Health Insurance Portability and Accountability Act (HIPAA): https://www.hhs.gov/hipaa/index.html
- Song, B., Li, F., Liu, Y., & Zeng, X. (2021). Deep learning methods for biomedical named entity recognition: A survey and qualitative comparison. *Briefings in Bioinformatics, 22*(6), 1-18. https://doi.org/10.1093/bib/bbab282
- Hu, Y., Chen, Q., Du, J., Peng, X., Keloth, V. K., Zuo, X., Zhou, Y., Li, Z., Jiang, X., Lu, Z., Roberts, K., & Xu, H. (2024). Improving large language models for clinical named entity recognition via prompt engineering. *Journal of the American Medical Informatics Association, 2024*(00), 1-10. https://doi.org/10.1093/jamia/ocad259

8. Appendices

Appendix A: Sample of App.py

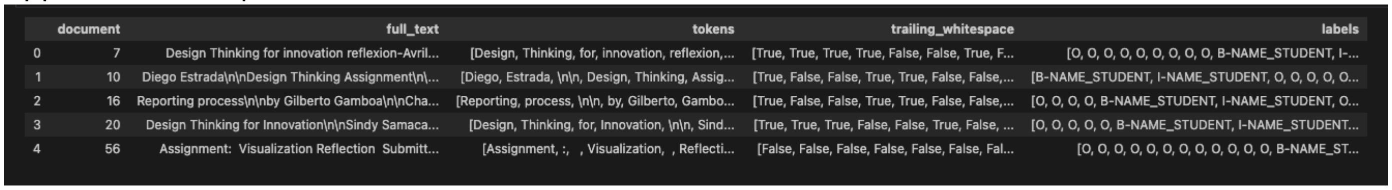
```
## App.py
## Deployment Backend
from flask import Flask, request, jsonify, send_file
from flask_cors import CORS
import spacy
from werkzeug.utils import secure_filename
import os
```

```python
import fitz  # PyMuPDF

app = Flask(__name__)
CORS(app)
nlp = spacy.load("custom_ner_model")

UPLOAD_FOLDER = 'uploads'
ALLOWED_EXTENSIONS = {'pdf', 'doc', 'docx'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

@app.route('/redact', methods=['POST'])
def redact_text():
    data = request.json
    text = data.get('text', '')
    doc = nlp(text)
    redacted_text = text

    for ent in doc.ents:
        redacted_text = redacted_text.replace(ent.text, f'[{ent.label_}]')

    return jsonify({'redacted_text': redacted_text})

@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return jsonify({"error": "No file part"}), 400

    file = request.files['file']
    if file.filename == '':
        return jsonify({"error": "No selected file"}), 400

    if file and allowed_file(file.filename):
        filename = secure_filename(file.filename)
        file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
        file.save(file_path)
        redacted_file_path = redact_pdf(file_path)
        return send_file(redacted_file_path, as_attachment=True)

    return jsonify({"error": "File type not allowed"}), 400

def redact_pdf(file_path):
    doc = fitz.open(file_path)
    for page in doc:
        text = page.get_text("text")
        redacted_text = text
        doc_nlp = nlp(text)
        for ent in doc_nlp.ents:
            redacted_text = redacted_text.replace(ent.text, f'[{ent.label_}]')
        page.insert_text((0, 0), redacted_text, fontsize=11, color=(0, 0, 0))
    redacted_file_path = file_path.replace('.pdf', '_redacted.pdf')
    doc.save(redacted_file_path)
    return redacted_file_path

if __name__ == '__main__':
    os.makedirs(UPLOAD_FOLDER, exist_ok=True)
    app.run(debug=True)
```

Appendix B: Sample of App.js

```
// App.js
// Code for Frontend of App.py
import React, { useState } from 'react';
import axios from 'axios';
import './App.css';

function App() {
    const [text, setText] = useState('');
    const [redactedText, setRedactedText] = useState('');
    const [file, setFile] = useState(null);

    const handleTextSubmit = async (e) => {
        e.preventDefault();
        try {
            const response = await axios.post('http://localhost:5000/redact', { text
});
            setRedactedText(response.data.redacted_text);
        } catch (error) {
            console.error('Error redacting text:', error);
            setRedactedText('Error redacting text. Please try again.');
        }
    };

    const handleFileSubmit = async (e) => {
        e.preventDefault();
        if (!file) {
            alert("Please upload a file.");
            return;
        }

        const formData = new FormData();
        formData.append('file', file);

        try {
            const response = await axios.post('http://localhost:5000/upload',
formData, {
                headers: {
                    'Content-Type': 'multipart/form-data'
                }
            });
            const url = window.URL.createObjectURL(new Blob([response.data]));
            const link = document.createElement('a');
            link.href = url;
            link.setAttribute('download', 'redacted.pdf');
            document.body.appendChild(link);
            link.click();
        } catch (error) {
            console.error('Error uploading file:', error);
            alert('Error uploading file. Please try again.');
        }
    };

    return (
        <div className="App">
            <h1>Text Redaction</h1>
            <form onSubmit={handleTextSubmit}>
                <textarea
                    value={text}
                    onChange={(e) => setText(e.target.value)}
                    placeholder="Enter text here"
                ></textarea>
                <button type="submit">Redact</button>
            </form>
            <h2>Redacted Text</h2>
            <div className="redacted-text">
                {redactedText ? <p>{redactedText}</p> : <p>No text redacted yet.</p>}
            </div>
            <h2>Upload PDF</h2>
            <form onSubmit={handleFileSubmit}>
                <input
                    type="file"
                    accept=".pdf, .doc, .docx"
                    onChange={(e) => setFile(e.target.files[0])}
                />
                <button type="submit">Upload and Redact</button>
            </form>
        </div>
    );
}

export default App;
```

Appendix C: Sample of Data



	document	full_text	tokens	trailing_whitespace	labels
0	7	Design Thinking for innovation reflexion-Avril...	[Design, Thinking, for, innovation, reflexion,...	[True, True, True, True, False, False, True, F...	[O, O, O, O, O, O, O, O, O, B-NAME_STUDENT, I-...
1	10	Diego Estrada\n\nDesign Thinking Assignment\n\...	[Diego, Estrada, \n\n, Design, Thinking, Assig...	[True, False, False, True, True, False, False,...	[B-NAME_STUDENT, I-NAME_STUDENT, O, O, O, O, O...
2	16	Reporting process\n\nby Gilberto Gamboa\n\nCha...	[Reporting, process, \n\n, by, Gilberto, Gambo...	[True, False, False, True, True, False, False,...	[O, O, O, O, B-NAME_STUDENT, I-NAME_STUDENT, O...
3	20	Design Thinking for Innovation\n\nSindy Samaca...	[Design, Thinking, for, Innovation, \n\n, Sind...	[True, True, True, False, False, True, False, ...	[O, O, O, O, O, B-NAME_STUDENT, I-NAME_STUDENT...
4	56	Assignment: Visualization Reflection Submitt...	[Assignment, :, , Visualization, , Reflecti...	[False, False, False, False, False, False, Fal...	[O, O, O, O, O, O, O, O, O, O, O, O, B-NAME_ST...