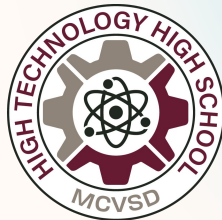


Machine Learning at Pfizer



Hillel Saks and Daniel D'Silva

2025 Spring Mentorship at High Technology High School

Instructor: Gabriel Guzman

Mentors: Lenny Grinberg, Jacob James Kerner, Christine Marie Gelotte

Pfizer History

- Founded in 1849 in Brooklyn by Charles Pfizer and Charles Erhart as a fine-chemicals company
- Rose to prominence during WWII by mass-producing penicillin
- Developed major drugs like Lipitor, Viagra, and Zoloft
- Expanded through acquisitions (e.g., Warner-Lambert, Wyeth, Hospira)
- Co-developed the first widely approved COVID-19 mRNA vaccine with BioNTech in 2020
- Global leader in pharmaceuticals
- Relies on machine learning and data analytics for drug discovery, clinical trial optimization, personalized medicine, etc.



Mentors

Lenny Grinberg, Director, WAN Engineering Discipline Lead


Set up, maintain, and test wide area networks.

Jacob Kerner, Machine Learning Engineer

Primarily focuses on natural language processing and generative AI.

Christine Gelotte, Cybersecurity Automation Engineer

Creates automated solutions to ensure security of Pfizer systems.



Redacting Sensitive Information in Dialogue and PDF Data

Building an NLP Pipeline with spaCy, PyMuPDF, and FaithDial



Background

- Large dialogue datasets contain sensitive content related to religion and ethnicity
 - Poses risks for both privacy and bias in NLP applications
 - Relevant to prioritizing patient privacy in medical documents
- Real-world documents need automated redaction workflows for compliance and ethics
- Standard PII (Personally Identifiable Information) removal models lack parameters of race, religion, and ethnicity

Project Goals

- Create a unified redaction pipeline for text files and PDFs containing PII
- Align original and redacted data for training privacy-respecting NLP models
- Develop a pipeline for redacting sensitive words in PDFs without distorting layout
- Ensure all outputs are consistent, verifiable, and reproducible



Fetching Religion and Ethnicity Tags

- Scraped names from Wikipedia using requests and BeautifulSoup
- Extracted ethnic group names from HTML tables and converted them into lowercase token patterns
- Formatted all religion and ethnicity terms into spaCy-compatible EntityRuler patterns
- Saved 1337 patterns (Religions: 88, Ethnicities: 1249)

```
[{"label": "RELIGION",  
  "pattern": [{"LOWER":  
    "buddhist"}]},  
{"label": "RELIGION",  
  "pattern": [{"LOWER":  
    "buddhists"}]},  
{"label": "RELIGION",  
  "pattern": [{"LOWER":  
    "catholic"}]},  
{"label": "RELIGION",  
  "pattern": [{"LOWER":  
    "catholics"}]}}
```

Scrubbing PII

- Used SpaCy with an EntityRuler loaded from JSON tags
- Replaced spans with corresponding placeholder text
 - Custom scrub() function
 - Used in other scripts to redact text

Sample text: My friend is an Orthodox Christian.

Output: My friend is an [RELIGION] [RELIGION].

```
def scrub(text):  
    doc = nlp(text)  
    spans = [(ent.start_char, ent.end_char,  
ent.label_) for ent in doc.ents if ent.label_ in  
{"RELIGION", "ETHNICITY"}]  
    redacted = text  
    offset = 0  
    for start, end, label in spans:  
        redacted = redacted[:start + offset] +  
f"[{label}]" + redacted[end + offset:]  
        offset += len(f"[{label}]") - (end -  
start) return redacted
```


Redacting sensitive fields in PDFs

- Load the PDF specified by pdf_path
- Processed each page of the PDF
 - Use PyMuPDF to modify content
- Extracted the raw text of each page
- Scrubbed the text using PII scrub function
- Detected differences between original and redacted text
- Built a list of words_to_redact by identifying which original words had been replaced
 - Located and redacted these words on the PDF page

- 1 -

SAMPLE

(All names and details provided in this sample are fictitious.
Some fields have been deliberately left blank.)

MEDICAL REPORT

SECTION 1: PATIENT'S PARTICULARS

Full name of patient: Mr Tan Ah Kow
NRIC/FIN/Passport no. of patient: S1111111X
Age of patient: 55 years old

SECTION 2: DOCTOR'S PARTICULARS

Full name of doctor: Tan Ah Moi
NRIC/FIN/Passport no. of doctor: S2222222Z
MCR no. of doctor: 333333
Hospital / Clinic name and address: 1 Blackacre Hospital, [REDACTED] 01010101
Doctor's qualifications and experience in this area of work: [To set out details]

Redacted sensitive field: Singapore

FaithDial dataset

Dataset summary

“FaithDial is a faithful knowledge-grounded dialogue benchmark, composed of 50,761 turns spanning 5649 conversations. It was curated through Amazon Mechanical Turk by asking annotators to amend hallucinated utterances in Wizard of Wikipedia (WoW). In our dialogue setting, we simulate interactions between two speakers...”

- Available on HuggingFace
- Contains discussions about faith, beliefs, worldviews etc.
 - Sufficient for training

```
{ "dialog_idx": 3, "response": "InRugby is
very popular in Northern England.\",
\"original_response\": \"InRugby is very
popular in England.\", \"history\": [
  \"\\nIt's almost the best time of year. the
first kickoff of American Football
season.\", \"\\nAnd to just think; the first
match of American Football was played on
November 6, 1869!\", \"\\nI'm sure it's a
complete different game then it is today.
The year the Detroit Lions will win the
super bowl.\", \"\\nThe first match was
between Rutgers and Princeton.\",
  \"\\nStrange now those are college teams and
not NFL!\", \"\\nThey were college teams
then too?\", \"\\nAny clue when the NFL
started?\", \"\\nI don't know. I do know
that the sport evolved from rugby and
association football.\", \"\\nI can see the
similarities. I love rugby. I wish it was
more popular here.\" ], \"knowledge\":
\"InRugby league is played across England but
is most popular in Northern England,
especially Yorkshire and Lancashire where
the game originated.\", \"BEGIN\": [
  \"Hallucination\" ], \"VRAM\": [ \"Edification\" ]
}
```

Redacting the FaithDial Dataset

- Applied custom scrub() function to:
 - Every utterance in the history list
 - Response, original_response, and knowledge fields
- Iterated over each data split
 - Train, test, validation
- Saved redacted items as new JSON files

```
dataset_path = "./faithdial_dataset"
dataset = load_from_disk(dataset_path)

for split in dataset.keys():
    print(f"Processing split: {split}")
    redacted_data = []

    for item in dataset[split]:
        if "history" in item and isinstance(item["history"], list):
            new_history = []
            for utterance in item["history"]:
                redacted = piiEntityLabel.scrub(utterance)
                new_history.append(redacted)
            item["history"] = new_history

        for field in ["response", "original_response", "knowledge"]:
            if field in item and isinstance(item[field], str):
                redacted = piiEntityLabel.scrub(item[field])
                item[field] = redacted

        redacted_data.append(item)

output_path = os.path.join(dataset_path, f"./redacted/{split}_redacted.json")
with open(output_path, "w", encoding="utf-8") as f:
    json.dump(redacted_data, f, indent=2, ensure_ascii=False)
```

Preparing SpaCy Data

- Loaded original and redacted versions of the FaithDial dataset splits
- For each pair of original and redacted texts:
 - Searched the redacted text for occurrences of redaction tags
 - Used context window around each redaction tag to locate the corresponding position in the original text
 - Calculated the start and end character indices of the replaced sensitive information
- Created spaCy Doc objects with original text and assigned the found entity spans
- Stored processed documents in a spaCy DocBin format suitable for training

```
First example with entities in split 'test':  
Original: And to just think; the first match of American football was played on November 6, 1869!  
Redacted:  
And to just think; the first match of [ETHNICITY] football was played on November 6, 1869!  
Entities: [(38, 46, 'ETHNICITY')]  
[TEST] Samples: 1823, Total Entities: 72  
Saved to: spacy_data\test.spacy
```

Running SpaCy Model

- Initialized a Blank English Pipeline
- Registered Entity Labels
- Trained the NER Model
 - Used spaCy's minibatch and compounding strategies

```
Loading training data...
Setting up blank NLP pipeline with NER component...
Registering entity labels...
Labels registered: ['ETHNICITY', 'RELIGION']
Starting training...
Iteration 1/20 - Losses: {'ner': np.float32(1673.7405)}
Iteration 2/20 - Losses: {'ner': np.float32(854.84674)}
Iteration 3/20 - Losses: {'ner': np.float32(690.2848)}
Iteration 4/20 - Losses: {'ner': np.float32(574.89233)}
Iteration 5/20 - Losses: {'ner': np.float32(492.94296)}
Iteration 6/20 - Losses: {'ner': np.float32(435.3651)}
Iteration 7/20 - Losses: {'ner': np.float32(337.11032)}
Iteration 8/20 - Losses: {'ner': np.float32(314.3436)}
Iteration 9/20 - Losses: {'ner': np.float32(283.96283)}
Iteration 10/20 - Losses: {'ner': np.float32(275.90167)}
Iteration 11/20 - Losses: {'ner': np.float32(253.51868)}
Iteration 12/20 - Losses: {'ner': np.float32(231.41933)}
Iteration 13/20 - Losses: {'ner': np.float32(196.51187)}
Iteration 14/20 - Losses: {'ner': np.float32(193.76917)}
Iteration 15/20 - Losses: {'ner': np.float32(182.16307)}
Iteration 16/20 - Losses: {'ner': np.float32(166.70602)}
Iteration 17/20 - Losses: {'ner': np.float32(157.16794)}
Iteration 18/20 - Losses: {'ner': np.float32(151.14833)}
Iteration 19/20 - Losses: {'ner': np.float32(137.80756)}
Iteration 20/20 - Losses: {'ner': np.float32(136.12651)}
- Model trained and saved to: ./pii_ner_model
```

```
log("Loading training data...")
doc_bin = DocBin().from_disk(TRAIN_DATA_PATH)
docs = list(doc_bin.get_docs(spacy.blank("en").vocab))

log("Setting up blank NLP pipeline with NER component...")
nlp = spacy.blank("en")
ner = nlp.add_pipe("ner")

log("Registering entity labels...")
labels = {ent.label_ for doc in docs for ent in doc.ents}
if not labels:
    raise ValueError("No entity labels found in training data.")
for label in labels:
    ner.add_label(label)
log(f"Labels registered: {sorted(labels)}")

log("Starting training...")
other_pipes = [pipe for pipe in nlp.pipe_names if pipe != "ner"]
with nlp.disable_pipes(*other_pipes):
    optimizer = nlp.begin_training()

    for i in range(N_ITER):
        random.shuffle(docs)
        losses = {}

        batches = minibatch(docs, size=compounding(BATCH_START, BATCH_END, BATCH_COMPOUND))
        for batch in batches:
            examples = [
                Example.from_dict(doc, {
                    "entities": [(ent.start_char, ent.end_char, ent.label_) for ent in doc.ents]
                }) for doc in batch
            ]
            nlp.update(examples, drop=DROPOUT_RATE, losses=losses, sgd=optimizer)

        log(f"Iteration {i + 1}/{N_ITER} - Losses: {losses}")

os.makedirs(OUTPUT_DIR, exist_ok=True)
nlp.to_disk(OUTPUT_DIR)
print(f"Model trained and saved to: {OUTPUT_DIR}")
```

Evaluating Model

- Loaded the test dataset
- Compared predicted entities (pred_ents) with the true entities (gold_ents) using character offsets and labels
- Recorded whether the entity was present in the ground truth (true_labels) and whether it was predicted (pred_labels)
- Used SkLearn to calculate evaluation metrics

```
true_labels = []
pred_labels = []

for doc in test_docs:
    pred_doc = nlp(doc.text)
    gold_ents = {(ent.start_char, ent.end_char, ent.label_) for ent in doc.ents}
    pred_ents = {(ent.start_char, ent.end_char, ent.label_) for ent in pred_doc.ents}

    all_offsets = gold_ents.union(pred_ents)
    for start, end, label in all_offsets:
        true_labels.append((start, end, label) in gold_ents)
        pred_labels.append((start, end, label) in pred_ents)

log("Evaluating...")
if not true_labels:
    print("No entities found in the test data. Cannot compute evaluation metrics.")
else:
    precision, recall, f1, _ = precision_recall_fscore_support(
        true_labels, pred_labels, average='binary'
    )
    print("Evaluation Results (TEST SET):")
    print(f"Precision: {precision:.3f}")
    print(f"Recall: {recall:.3f}")
    print(f"F1 Score: {f1:.3f}")
```

Results

Text: And to just think; the first match of American football was played on November 6, 1869!

Predicted Entities:

American -> ETHNICITY [38:46]

Text: Well, after the American Revolutionary War, the students experienced harsh treatments from teachers.

Predicted Entities:

American -> ETHNICITY [16:24]

Text: No, I have never. Speaking of Irish ancestry, would you say you are a Irish American, meaning that you identify with the ancestry ?

Predicted Entities:

Irish -> ETHNICITY [30:35]

Evaluation Results (Test Set)

- Precision: 0.652
- Recall: 0.662
- F1 Score: 0.656

Thank you for listening!

