Innocence Discovery Lab - Harnessing Large Language Models to Surface Data Buried in Wrongful Conviction Case Documents

Ayyub Ibrahim
Innocence Project New Orleans
U.S.A.

Huy Dao
Innocence Project New Orleans
U.S.A.

Tarak Shah
Human Rights Data Analysis Group
San Francisco, CA
U.S.A.

The recent advent of commercial artificial intelligence (AI), especially in natural language processing (NLP), introduces transformative possibilities for wrongful conviction research. NLP, a pivotal branch of AI that forms the basis for Large Language Models (LLMs), enables computers to interpret human language with a nuanced understanding. This technological advancement is particularly valuable for analyzing the complex language found in case documents associated with wrongful convictions. This paper explores the effectiveness of LLMs in analyzing and extracting data from case documents collected by the Innocence Project New Orleans and the National Registry of Exonerations. The diverse and comprehensive nature of these datasets makes them ideal for assessing the capabilities of LLMs. The findings of this study advance our understanding of how LLMs can be utilized to make wrongful conviction case documents easily accessible by automating the extraction of relevant data.

I Introduction

Thousands of wrongfully convicted people have been exonerated across the globe in the last four decades. These wrongful convictions have not only uncovered and undone mistakes in investigations and prosecutions, they have exposed profound and systemic injustice in criminal legal systems themselves. The patterns and practices that inform - indeed create - wrongful convictions have largely remained hidden. They often start with how communities are policed, from the moment a crime is reported or observed. As exonerations continue to happen, a growing body of documents and data remains untapped. They have the potential to reveal not only the identities of law enforcement involved in wrongful convictions, but also their roles in each case, their patterns of misconduct and migration, and how they are connected not only to each other but to wrongful conviction cases not yet found or investigated. The Innocence Discovery Lab, born of Innocence Project New Orleans (IPNO) and the Louisiana Law Enforcement Accountability Database (LLEAD), seeks to leverage the advent of large language models to transform unstructured documents from case documents into a structured, accessible format.

Historically, the expansive volume of documentation associated with exoneration cases has presented significant analytical challenges. The sheer quantity and complexity of data, ranging from legal transcripts and police reports to witness statements and forensic analyses, have long made it difficult to extract meaningful insights and make important connections. These difficulties are compounded by the diverse nature of the documents, which often include varying formats and levels of detail. Large language models, like ChatGPT, now make processing these documents at scale feasible due to their ability to process and interpret vast amounts of text. These capabilities include analyzing the language used in legal documents to detect biases, examining patterns in policing practices, and cross-referencing details across cases to identify systemic issues. Furthermore, the ability of these models to learn and adapt over time means that they become more efficient and accurate as they process more data, continuously enhancing their analytical power.

Key to our research is the integration of wrongful conviction data with police databases, including IPNO's internal database developed by its case management team, and its public counterpart, the Louisiana Law Enforcement Accountability Database (LLEAD). IPNO’s internal database, which houses case data from IPNO’s clients and applicants, offers detailed insights into individual wrongful conviction cases. LLEAD, with data on over 100,000 officers in Louisiana, provides a broader view of police misconduct and migration patterns. The combination of these databases enables a more comprehensive analysis that connects the specifics of individual cases to systemic issues in law enforcement. By correlating data from these two sources, our research aims not only to reveal specific links between wrongful convictions and law enforcement misconduct but also to shed light on broader trends and practices contributing to these injustices.

II Defining a Wrongful Conviction

Within the scope of our research, a 'wrongful conviction' is defined as the conviction of an individual who is factually innocent of the crime charged. This can result from a trial verdict or a plea. An exoneration is the official overturning of a wrongful conviction. These typically occur

through pardons or acquittals at retrial, often initiated by the emergence of new evidence that proves innocence and was not available or presented during the trial phase of the case.

III Framework for Exoneration Document Analysis

In Orleans Parish, Louisiana, where Innocence Project New Orleans is based, 78%[1] of wrongful convictions have been linked to law enforcement's failure to share exculpatory evidence with the defense, a rate more than double the national average.

Our research, recognizing the explicit relationship between law enforcement misconduct and wrongful convictions, aims to establish best practices for transforming unstructured wrongful conviction case data into structured, accessible formats. This transformation is crucial for lawyers, advocates, and community members who are committed to leveraging insights from past wrongful convictions to prevent future occurrences. Our methodology is built on a multi-stage process:

A. Metadata Compilation

The foundation of our research involves compiling metadata into an index, essential for effectively managing our extensive and growing corpus of exoneration documents. In organizing the metadata, we have focused on collecting details crucial for document identification and management. This includes capturing the file path and name, file type, file size, number of pages, and creating a unique identifier for each document by truncating the SHA1 content hash. A case ID is also assigned to each document, derived from the directory names used during the scanning process.

B. Page Classification

In the course of an exoneration case, a wide variety of documents are accumulated, reflecting materials that may extend over many decades. These documents are not only extensive in volume but also varied in nature, each offering a unique lens into the intricacies of the case. A challenge we frequently encounter is the presence of inaccurately or inconsistently named files, which prevents the immediate identification of their contents. Moreover, it is not uncommon to find lengthy documents, e.g. exceeding a thousand pages, containing a collection of different document types.

After consultation with the IPNO's case management team, we decided to focus on a specific subset of documents considered most relevant to wrongful conviction research. These documents included police reports, court transcripts, and court testimonies. To accurately classify the array of documents produced over the course of an exoneration case, we have developed an automated page classification model to overcome the limitations of traditional manual review methods. This model utilizes a pre-trained convolutional neural network, optimized through training on thumbnail images of key document types. The thumbnails' smaller sizes ensure

[1] Samuel R Gross *et al*, "Government Misconduct and Convicting the Innocent: The Role of Prosecutors, Police and Other Law Enforcement" (1 Sept 2020) online (pdf): Online: <law.umich.edu/special/exoneration/Documents/Government_Misconduct_and_Convicting_the_Innocent.pdf>.

efficient processing and minimal computational load. We used the FastAI library to adapt the ResNet34 architecture, initially trained on the ImageNet[2] database, for the identification of these document types from their thumbnails. This approach significantly streamlines the document classification process, overcoming the challenges posed by the volume and diversity of the documents.

C. Unstructured Data Extraction with Large Language Models

In the current stage of our research, which is the primary focus of this paper, we are constructing a database from unstructured text found within over 300,000 pages of exoneration documents collected from IPNO and the National Registry of Exonerations[3]. Our objective in creating this database is to index the identities of all police officers, prosecutors, and laboratory personnel featured in these documents, along with extracting detailed information about their actions and the events they were involved in. To facilitate this extraction, we are utilizing large language models for their advanced capabilities in parsing and interpreting complex text structures. This approach is particularly crucial given the diverse range of document types typically encountered in wrongful conviction cases. After the information is extracted, it will be converted into formats optimized for analysis. This structuring is expected to significantly enhance the accessibility and utility of the data, enabling more rigorous research into wrongful convictions.

D. Deduplication

After the structured data extraction, a significant challenge we face is the extensive issue of data redundancy, exacerbated by the nature of our dataset. Officers involved in wrongful convictions are often mentioned in various documents that span different periods of time. This results in instances where the same individual may appear with different names or in different contexts, leading to multiple entries in our initial index.

To address this issue, our approach involves filtering this index down to a table of unique identities. This process is designed to accurately identify and consolidate instances where the same officer is referenced in disparate capacities or times within the dataset. By implementing this filtration, we aim to effectively remove duplicate entries, thus preserving the accuracy and integrity of our dataset. This step is essential to ensure that the representation of each officer's involvement in wrongful convictions is consistent and precise in our analysis, thereby facilitating a clearer and more comprehensive understanding of each officer's role within a case.

E. Cross-referencing

Our research will culminate in a comparison of data from our exoneration case documents, soon to be indexed into a wrongful conviction database, with the Louisiana Law Enforcement Accountability Database (LLEAD) and the Innocence Project New Orleans' (IPNO) internal database. LLEAD, with data on approximately 100,000 police officers in Louisiana, including

---

[2] Kaiming He *et al*, "Deep Residual Learning for Image Recognition" (2015) ArXiv 1512.03385, online (pdf): <arxiv.org/pdf/1512.03385.pdf>

[3] University of Michigan Law School, "Spread-Sheet Request Form" (last accessed 5 Dec 2023), online: <law.umich.edu/special/exoneration/Pages/Spread-Sheet-Request-Form.aspx>.

records of over 40,000 misconduct allegations, will be a key resource. We will focus on identifying officers involved in both misconduct and wrongful convictions by analyzing data such as names, ranks, department affiliations, hire dates, and departure dates.

After initially cross-referencing our wrongful conviction data with the Louisiana Law Enforcement Accountability Database (LLEAD), we will proceed to a further stage of analysis that involves cross-referencing the officers we've identified as having histories of misconduct and associations with wrongful convictions with the Innocence Project New Orleans' (IPNO) internal database. This internal database is particularly significant, as it contains names of officers flagged during case review and by potential clients.

Through this additional layer of cross-referencing, our objective is to deepen our understanding of each officer's potential involvement in wrongful convictions, past and potentially future. This comprehensive approach, which includes cross-referencing with both the LLEAD and IPNO's internal database, will guide IPNO's case management team in scrutinizing cases involving officers with direct ties to wrongful convictions as well as those indirectly connected, such as their partners, supervisors, departments/divisions they've moved to, or their trainees. A particular focus will be placed on those associated officers with a substantial history of misconduct or identified involvement in wrongful conviction cases.

F. Structured Data Extraction with Regular Expressions (Regex)

Our research into wrongful convictions begins with the extraction of structured, meaningful data from a vast store of documents containing unstructured text. While our current research focuses on employing large language models (LLMs) for this task, it's essential to acknowledge the various methodologies that have historically shaped Information Extraction (IE) in Natural Language Processing (NLP), a field that has experienced significant growth over the past few decades[4] in response to the growth in complexity and volume of data.

The earliest iterations of IE were predominantly rule-based and dictionary-based systems. These initial methods involved the application of manually created rules and the use of curated dictionaries for phrase matching. While capable of high accuracy in small and well-defined datasets, these systems lacked flexibility and scalability. The requirement for extensive manual input to develop rules and dictionaries rendered them less practical for large or dynamically changing datasets. Moreover, the language specificity of these methods restricted their effectiveness across different linguistic contexts.

With the general increase in data availability and complexity, the field of IE gravitated towards statistical machine learning methods. This shift responded to the limitations of rule-based systems, as these newer methods utilized algorithms that could learn directly from data, moving beyond the confines of manually programmed rules. Techniques like Hidden Markov Models (HMM), Maximum Entropy Models (MEM), Support Vector Machines (SVM), and Conditional Random Fields (CRF) were increasingly employed to extract statistical features from manually

[4] Yang Yang *et al*, "A Survey of Information Extraction Based on Deep Learning" (2022) 12:19 Applied Sciences 9691, online: <mdpi.com/2076-3417/12/19/9691>.

labeled corpora. This advancement in scalability came with its own challenges, most notably the need for manual annotation and complex feature engineering.

In the evolving landscape of IE, deep learning methodologies represent the latest advancement. Models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have revolutionized IE with their proficiency in feature extraction and ability to learn from large datasets. Automating the feature engineering process, these models are adept at identifying complex patterns in data, significantly enhancing fields like NLP, speech recognition, and visual object recognition. However, despite their groundbreaking potential, deep learning models, including advanced large language models (LLMs), are not without limitations. They require substantial datasets for training and significant computational resources, presenting barriers in certain contexts. This contrast between the capabilities of deep learning and the practical constraints of various IE methods, from rule-based to deep learning approaches, highlights the considerations in selecting an appropriate technique for specific research challenges.

Acknowledging the range of information extraction techniques, we opted for a rules-based approach for our baseline model, specifically regex, due to its immediate deployability and efficiency in pattern recognition.

The primary task of extracting officer information, however, quickly highlighted the limitations of regex when handling the complexities of natural language found in wrongful conviction case documents. Consider the challenges posed by a complex sentence from a court transcript: "John Ruiz was mentioned as being involved in the joint investigation with Detective Martin Sholtz regarding the Seafood City burglary and the murder of Randy Gray." In this scenario, regex's capabilities are notably limited. While regex can effectively identify 'Detective Martin Sholtz' as a key entity, owing to the clear pattern of a recognized title followed by a name, it may fail to recognize 'John Ruiz' in a similar capacity. This limitation arises because regex operates on predefined patterns and lacks the ability to understand semantic nuances. In our example, regex is programmed to detect titles like 'Detective' followed by names, which it does successfully for 'Detective Martin Scholtz'. However, without the explicit title 'Detective' preceding 'John Ruiz', regex overlooks this name, despite its relevance in the narrative. This shortfall illustrates a critical aspect of regex's nature: its inability to infer context or understand relational connections between entities in text. Consequently, important information like the involvement of John Ruiz in the investigation, equally crucial to the case narrative, might be missed, underscoring the need for more sophisticated methods capable of semantic comprehension in legal text analysis.

To demonstrate the limitations of regex, we created a baseline regex model designed to extract officer names. This model’s intent was not to comprehensively extract all details related to officers but to evaluate the effectiveness of regex in extracting a specific type of structured data. The model, captured in the pattern below, was tested on police reports and court transcripts.

Listing 1.0: Regular Expression Pattern

```
pattern = .compile(r”(detective|sergeant|lieutenant|captain|corporal|deputy|
investigator|criminalist|technician|det\.|sgt\.|lt\.|cpt\.|cpl\.|dty\.|tech\.|dr\.)\s+([A-Z][A-Za-
z]*(\s[A-Z][A-Za-z]*)?)", re.IGNORECASE)
```

Evaluating the performance of our regex model involved analyzing metrics such as precision, recall, F1 score, and F-beta score. Precision, which measures the accuracy of the model in correctly identifying officer names, was found to be 84.5% in police reports, suggesting a high level of reliability in the model's identifications.

However, recall, which assesses the model's ability to detect all relevant instances of officer names, was only 51.8% in police reports. This indicates that the model missed almost half of the actual officer names present in the documents. The F1 score, a metric combining precision and recall, stood at 0.614, reflecting a moderate balance between these two aspects.

We also considered the F-beta score, a variation of the F1 score that gives more weight to recall. Given the critical nature of not missing any true positives in our research, recall was weighted more heavily. In the context of police reports, the F-beta score was 0.549, highlighting the model's limitations in recall.

The model's performance in court transcripts followed a similar trend but with more noticeable shortfalls. It achieved a precision of 86.56%, signifying relatively accurate identifications. However, its recall dropped to 42.81%, indicating that the model failed to detect more than half of the officer names in these documents. Consequently, the F1 score was recorded at 0.5461, and the F-beta score, prioritizing recall, further decreased to 0.4663.

Figure 1. Baseline Regular Expression Model Evaluation

Metric	Police Reports	Court Transcripts
Precision	0.845	0.8656
Recall	0.518	0.4281
F1 Score	0.614	0.5461
F-Beta	0.549	0.4663

The overall performance points to the inherent limitations of regex in handling the nuanced and context-rich language of legal documents. The low F-Beta score, particularly in court transcripts, emphasizes the need for more sophisticated data extraction techniques that can capture the full range of necessary information.

The challenges we encountered with regex highlighted the need for more advanced data extraction methods, specifically those capable of comprehending and interpreting the semantic context within wrongful conviction case documents. This realization has steered our research towards the integration of large language models with regex. Large language models, with their advanced capabilities in understanding natural language, offer a promising solution for the complexities we face in extracting structured data from legal texts. By combining the pattern-

matching strengths of regex with the deep learning and contextual understanding[5] of large language models, we aim to significantly enhance our recall capabilities.

G. Structured Data Extraction with a Large Language Model

The process of structured data extraction using large language models (LLMs) presented a unique set of challenges, particularly in managing the substantial length of the documents we are analyzing. Single documents often extend to hundreds of pages, posing a significant contrast to the prompt length constraints of LLMs.

To effectively utilize LLMs in our data extraction process, we developed a strategy to isolate specific text segments within each document. These segments were then used to create more focused and efficient prompts for the language model.

We approached this task by identifying the relevant text segments, and secondly, by extracting structured information about officers from these identified segments. For managing this two-step process, we employed Langchain[6], a natural language processing library, and OpenAI's GPT.

For the first step, identifying the relevant chunks of text within the larger document, we used the approach outlined in Precise Zero-Shot Dense Retrieval without Relevance Labels[7]. This approach splits our information retrieval task into multiple steps:

1. A query requesting names and roles of mentioned officers was entered into a large language model, which then generated a "hypothetical" document in response.
2. This hypothetical document was embedded.
3. The document text was divided into overlapping chunks, with each chunk receiving an embedding using the same system as the hypothetical document.
4. Facebook's AI Similarity Search (FAISS)[8], a nearest-neighbor search implementation, was then used to identify relevant text content by comparing chunk embeddings with those of the hypothetical document.

H. Hypothetical Document Embeddings

Hypothetical Document Embeddings (HyDE) transform raw text into a structured, searchable format. This process begins with a large language model generating a hypothetical

---

[5] Somin Wadhwa, Silvio Amir & Byron C Wallace, "Revisiting Relation Extraction in the era of Large Language Models" (2023) 1 Proc 61st Annual Meeting of the Association for Computational Linguistics 15566, online (pdf): <aclanthology.org/2023.acl-long.868.pdf>.

[6] LangChain, "Get Started" (last accessed Dec 5, 2023) online: <python.langchain.com/docs/get_started/introduction>

[7] Gao *et al*, "Precise Zero-Shot Dense Retrieval without Relevance Labels" (2022) ArXiv:2212.10496, online: <arxiv.org/abs/2212.10496>.

[8] Hervé Jegou, Matthijs Douze & Jeff Johnson, "Facebook AI Similarity Search (FAISS)" (29 Mar 2017) Engineering at Meta, online: <engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/>.

document in response to a query. The strength of this document lies in its pattern-rich content, essential for locating documents with similar content in a database, despite potential inaccuracies.

The next step involves converting the hypothetical document into an embedding vector. This conversion translates the text into vector representations in a multi-dimensional space. These embeddings capture more than simple word counts or keyword matches; they encapsulate the text's contextual nuances and underlying intent. Thus, searches leveraging these embeddings focus on contextual similarity and semantic connections between documents, surpassing traditional keyword-based search methods in depth and relevance.

Listing 2.0: Hypothetical Document Embeddings Query

```
PROMPT_TEMPLATE_HYDE = PromptTemplate(input_variables=["question"],
template="""
You're an AI assistant specializing in criminal justice research.
Your main focus is on identifying the names and providing detailed context of mention for
each law enforcement personnel.

This includes police officers, detectives, deputies, lieutenants, sergeants, captains,
technicians, coroners, investigators, patrolmen, and criminalists, as described in court
transcripts and police reports.

Question: {question}

Responses:
""",
)
```

Listing 2.1: Hypothetical Document Embeddings Implementation

```
def generate_hypothetical_embeddings():
llm = OpenAI()
prompt = PROMPT_TEMPLATE_HYDE
llm_chain = LLMChain(llm=llm, prompt=prompt)
base_embeddings = OpenAIEmbeddings()
embeddings = HypotheticalDocumentEmbedder(
llm_chain=llm_chain, base_embeddings=base_embeddings
)
return embeddings
```

To create the vector database, we utilize our ‘process_single_document’ function. This function initiates by loading the text of a document and segmenting it. For segmentation, we use LangChain’s RecursiveCharacterTextSplitter, which divides the document into word chunks. The chunk size and overlap are chosen to ensure that each segment is comprehensive enough to maintain context while being sufficiently small for efficient processing. Post-segmentation, these chunks are transformed into high-dimensional vectors using the hypothetical document's embedding scheme. The concluding step involves the ‘FAISS.from_documents function’, which compiles these vectors into an indexed database. This database enables efficient and context-sensitive searches, allowing for the quick identification of documents that share content similarities with the hypothetical document.

Listing 3.0: **Storing the Document in a Vector Database**

```
def process_single_document(file_path, embeddings):
logger.info(f"Processing document: {file_path}"
loader = JSONLoader(file_path)
text = loader.load()
logger.info(f"Text loaded from document: {file_path}")
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500,
chunk_overlap=250)
docs = text_splitter.split_documents(text)
db = FAISS.from_documents(docs, embeddings)
return db
```

Following the creation of our vector database, the document becomes fit for structured information extraction. This task is carried out by the get_response_from_query function, which is designed to transform pre-processed, unstructured data into structured outputs.

Initial Query Processing: The extraction phase begins when a user sends a query to the vector database. Once the query is received, the database conducts a search within its embedding space, identifying and retrieving text chunks that best match the query's contextual and semantic criteria. This retrieval process is carried out using the ‘db.similarity_search_with_score’ method, which selects the top 'k' relevant chunks based on their high similarity to the query.

Sorting of Retrieved Chunks: After their retrieval, the chunks are sorted according to relevance using the ‘sort_retrieved_documents’ function. This step ensures that the most relevant chunks are appropriately organized within the model’s context window. This approach is supported by findings from 'Lost in the Middle: How Language Models Use Long Contexts,[9] which emphasize that language models typically yield better performance when pertinent information is positioned at the beginning or end of their input contexts. After sorting, the chunks are concatenated into a single string, eliminating the overhead of processing multiple individual strings and reducing unnecessary tokens.

Model Initialization and Response Generation: The processing begins with the instantiation of an OpenAI model and the LLMChain class. This setup allows the chain to process the combined document content along with the original query. Following this, the LLMChain executes its run method, using the inputs of prompt, query, and document content to generate a structured and detailed response. The model then extracts information relevant to the query and structures the output according to the specifications in the prompt template.

Listing 4.0: Template for Model

```
PROMPT_TEMPLATE_MODEL = PromptTemplate(input_variables=["question",
"docs"],template=""
```

[9] Nelson F Liu *et al*, “Lost in the Middle: How Language Models Use Long Contexts” (2023) ArXiv:2304.03173, online: <arxiv.org/abs/2307.03172>.

As an AI assistant, my role is to meticulously analyze criminal justice documents and extract information about law enforcement personnel.

Query: {question}
Documents: {docs}

The response will contain:

1) The name of a police officer.
Please prefix the name with "Officer Name: ".
For example, "Officer Name: John Smith".

2) If available, provide an in-depth description of the context of their mention.
If the context induces ambiguity regarding the individuals role in law enforcement, note this.
Please prefix this information with "Officer Context: ".

3) Review the context to discern the role of the officer. For example, Lead Detective.
Please prefix this information with "Officer Role: "
For example, "Officer Role: Lead Detective"

The full response should follow the format below, with no prefixes such as 1., 2., 3., a., b., c.:

Officer Name: John Smith
Officer Context: Mentioned as officer at the scene of the incident.
Officer Role: Patrol Officer

Officer Name:
Officer Context:
Officer Role:

Additional guidelines:
Only derive responses from factual information found within the police reports.
""",)

Listing 4.1: Function for Generating Responses

```
def get_response_from_query(db, query):
# Set up the parameters
prompt = PROMPT_TEMPLATE_MODEL
roles = ROLE_TEMPLATE
temperature = 1
k = 20

# Perform the similarity search
doc_list = db.similarity_search_with_score(query, k=k)

# Sort documents by relevance scores as suggested in the literature
```

```
docs = sorted(doc_list, key=lambda x: x[1], reverse=True)

third = len(docs) // 3
highest_third = docs[:third]
middle_third = docs[third:2*third]
lowest_third = docs[2*third:]
highest_third = sorted(highest_third, key=lambda x: x[1],reverse=True)
middle_third = sorted(middle_third, key=lambda x: x[1], reverse=True)
lowest_third = sorted(lowest_third, key=lambda x: x[1], reverse=True)
sorted_docs = highest_third + lowest_third + middle_third
# Join documents into one string for processing
docs_page_content = " ".join([d[0].page_content for d in sortedocs])

# Create an instance of the OpenAI model
llm = ChatOpenAI(model_name="gpt-4")

# Create an instance of the LLMChain
chain = LLMChain(llm=llm, prompt=prompt)

# Run the LLMChain and print the response
response = chain.run(question=query, docs=docs_page_content,
temperature=temperature)
print(response)

return response
```

I. Example of Model's Workflow

Query Input:

"In the transcript, identify individuals by their names along with their specific law enforcement titles, such as officer, sergeant, lieutenant, captain, commander, sheriff, deputy, detective, inspector, technician, analyst, and coroner. Alongside each name and title, note the context of their mention. This includes the roles they played in key events, decisions they made, actions they took, their interactions with others, responsibilities in the case, and any significant outcomes or incidents they were involved in."

Document Retrieved from the Vector Database:
(1 of 20 documents identified by the Faiss similarity search as relevant)
"Mark MeCrarey, New Orleans police sergeant. A 16 .01 Sergeant DeHarmen Cade, where are you assigned now? - A Second Police District. 13. And in October, September of 1979 and in Q 19 September and October of 1980, where were you assigned? :1 Homicide division. A. And how long have you been on the police department right now? Thirteen and a half years. A Dwight Fernandez, when did you or did you ever take over the investigation of ... murder? A", metadata={'source': '../../data/convictions/transcripts/iterative\\(C) Det. Martin Scholtz Testimony - Trial One.docx, 'page_number': 150}'"

Figure 2. Model's Response

Officer Name	Officer Context	Officer Role
Mark MeCrarey	Mentioned as the patrol officer of CAR 802 at the scene in the Garden district.	Patrol Officer
DeHerman Cade	DeHerman Cade is referenced multiple times throughout the transcript, indicating his active involvement in various aspects of the case. His roles include being assigned to the homicide division, where he appears to play a lead role in the investigation. The transcript details instances where he is actively engaging in investigative duties, analyzing evidence, and interacting with other law enforcement personnel.	Investigating Detective
Dwight Fernandez	Mentioned as being an Officer (from Upper Quarter Crime Prevention on detail) in various contexts, including when he was at the homicide office with Cade.	Officer

J. Performance Evaluation

To assess the model's effectiveness in extracting officer names from wrongful conviction case documents, we developed a detailed evaluation framework using the GPT-4 model. The framework was structured as a series of tests where the same query was executed six times to evaluate the consistency and reliability of the model. These tests were influenced by several key parameters, including preprocessing parameters like chunk size, which determines the volume of consecutive text units processed, and chunk overlap, indicating the number of shared words between consecutive text chunks. Additionally, the evaluation considered model-specific parameters such as the impact of Hypothetical Document Embeddings (HYDE) on model effectiveness, the 'k' value specifying the number of text chunks per query, and the temperature parameter, which controls the variability of outputs.

K. Preprocessing Parameters

Chunk Size: Determines the volume of consecutive text units processed in each instance.
Chunk Overlap: Indicates the number of words shared between consecutive text chunks. For example, a 250-word overlap means the subsequent chunk initiates 250 words before the end of the preceding one.

L. Model-Specific Parameters

Hypothetical Document Embeddings (HYDE): Their influence on the model's overall effectiveness was assessed.
'k' Value: Specifies the number of text chunks inputted into the model for each query.

Temperature Parameter: Controls the degree of variability in the model's outputFor the evaluation of our model, we chose the F-beta score as the primary metric because of its capacity

to balance and differentially weigh precision and recall. In our model's context, we prioritized recall, assigning it twice the importance of precision. This decision is in line with our goal to achieve thorough identification of relevant data, accepting the possibility of occasionally including some irrelevant information. Such an approach is particularly valuable in scenarios where missing key information is more critical than avoiding irrelevant data.

Our model achieved optimal performance with the following parameters: a chunk size of 500 words, a chunk overlap of 250 words, the integration of HYDE embeddings, and a 'k' value of 20. Utilizing these parameters, the model attained an F-beta score of 0.864909 for police reports and 0.813397 for court transcripts. Notably, larger chunk sizes (1000 and 2000 words) and greater overlaps (500 and 1000 words) reduced the F-beta score, despite providing more contextual information. HYDE embeddings consistently enhanced performance, proving vital to the model's effectiveness. Additionally, a temperature setting of 1 generally improved F-beta scores. However, it is crucial to manage this parameter carefully for extracting accurate officer context mentions in subsequent phases. Higher temperature settings, while increasing variability, risk generating inaccurate or fabricated content, an issue often termed 'hallucination' in language models.

Figure 3: Model Performance Evaluation

chunk_size	chunk_overlap	temperature	k	hyde	filetype	FN	FP	TP	n_files	precision	recall	F1	F-beta
500	250	1	20	1	transcript	3	27	34	4	0.557	0.919	0.694	0.813
500	250	0	20	1	report	6	56	60	5	0.517	0.909	0.659	0.789
2000	1000	1	5	0	report	12	32	71	5	0.689	0.855	0.763	0.816
2000	1000	1	5	1	transcript	3	11	17	3	0.607	0.85	0.708	0.787
500	250	1	20	1	report	20	2	105	5	0.981	0.84	0.905	0.865
1000	500	0	10	1	report	13	70	61	5	0.466	0.824	0.595	0.714
1000	500	0	10	1	transcript	15	31	57	6	0.648	0.792	0.712	0.758
2000	1000	0	5	1	report	13	37	49	5	0.57	0.79	0.662	0.734
2000	1000	0	5	0	report	15	13	54	5	0.806	0.783	0.794	0.787
500	250	0	20	1	transcript	19	29	53	6	0.646	0.736	0.688	0.716
2000	1000	0	5	1	transcript	22	18	60	7	0.769	0.732	0.75	0.739
2000	1000	1	5	1	report	37	19	86	5	0.819	0.699	0.754	0.72
1000	500	1	10	1	report	34	10	78	5	0.886	0.696	0.78	0.728
1000	500	1	10	1	transcript	16	32	19	4	0.373	0.543	0.442	0.497
2000	1000	0	5	0	transcript	44	36	50	9	0.581	0.532	0.556	0.541

Following the initial evaluations based on varying parameters, we continued to measure the model's performance through multiple iterative runs by employing two distinct query strategies. Given the inherent stochastic nature of large language models, which often yield varied outputs when processing the same document multiple times, these strategies were designed to assess the model's performance under static parameters. These additional tests were conducted using three models: GPT-4, GPT-3.5-Turbo-16K, and GPT-3.5-Turbo-4K.

The first strategy involved using six different queries, each specifically crafted to extract distinct types of information. This approach enabled us to evaluate the model's versatility in handling a variety of information extraction tasks. The second strategy used a single, comprehensive query, designed to extract all relevant information, over six iterations. This approach enabled us to measure the model's performance across successive iterations.

Figure 4. Average Cumulative F-Beta Score by Iteration for Police Reports GPT-4 (Standard Model)

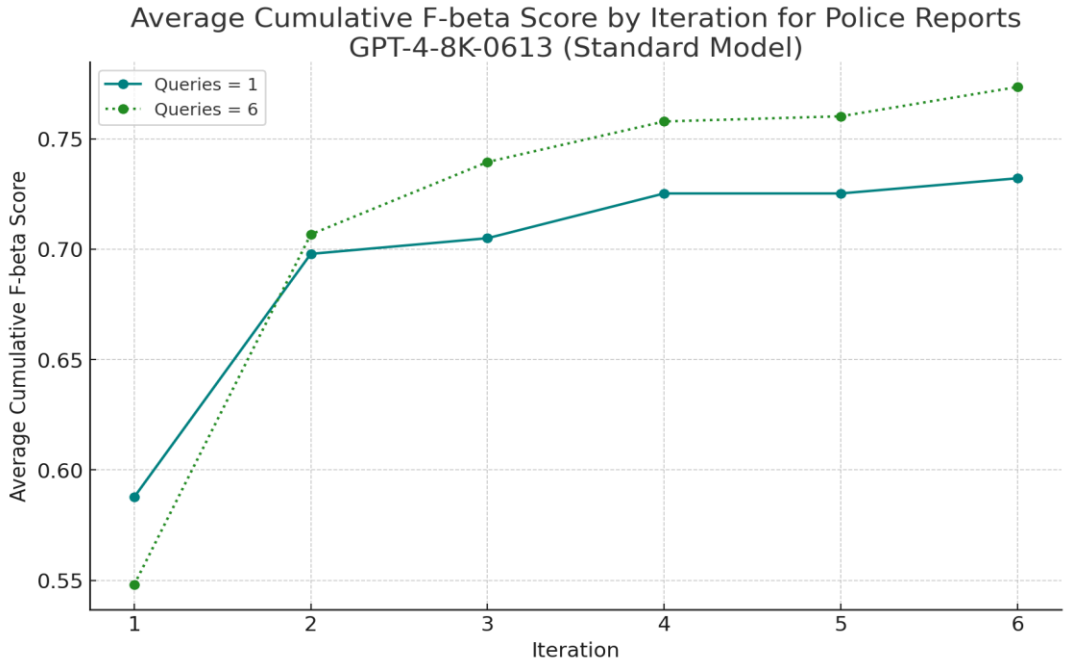


Figure 5. Average Cumulative F-Beta Score by Iteration for Transcripts Reports GPT-4 (Standard Model)

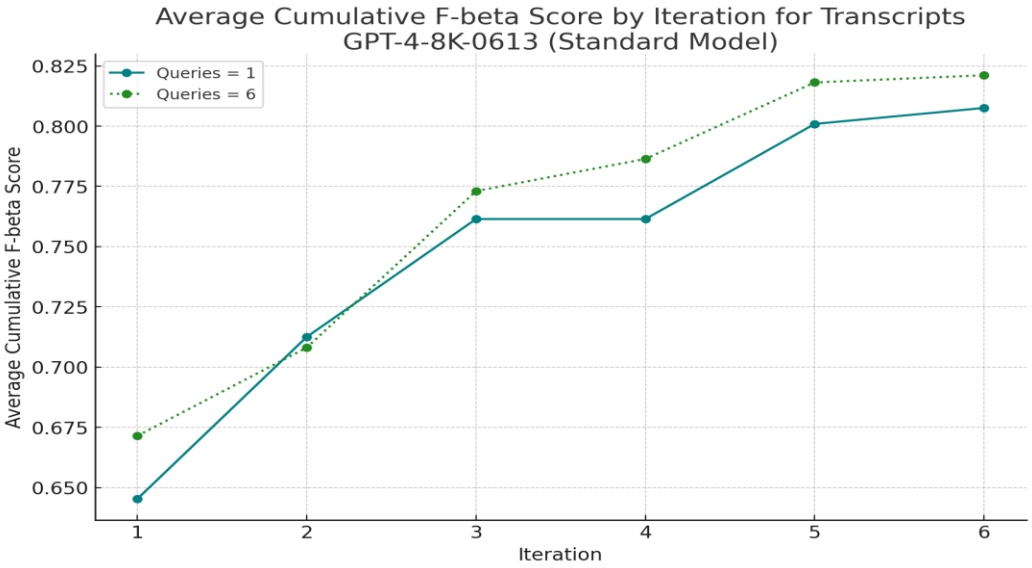


Figure 6. Average Cumulative F-Beta Score by Iteration for Police Reports GPT-3.5-Turbo-16K-0613 (Standard Model)

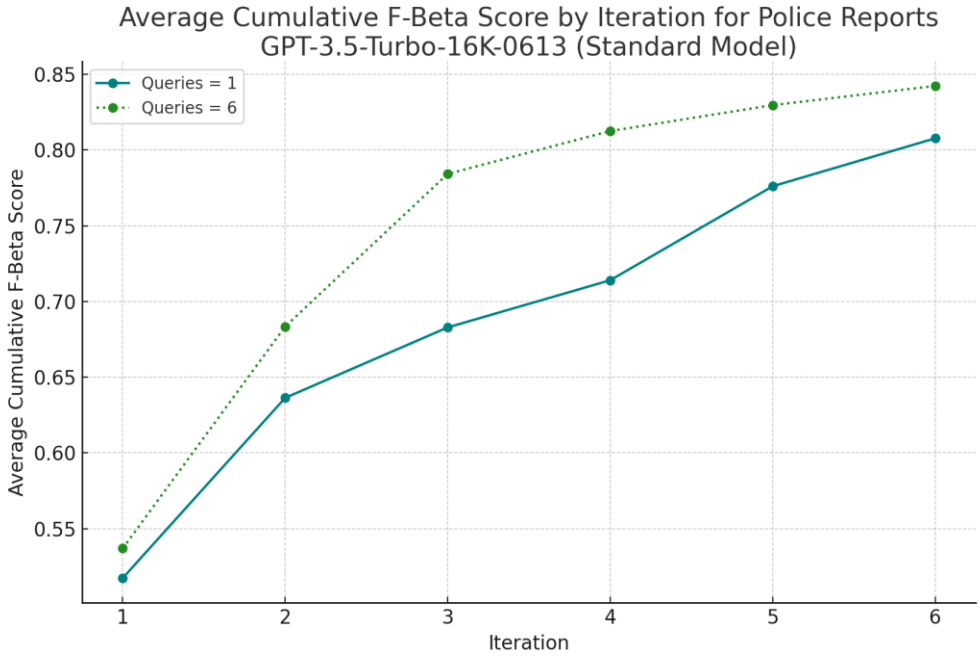


Figure 7. Average Cumulative F-Beta Score by Iteration for Transcripts GPT-3.5-Turbo-16K-0613 (Standard Model)

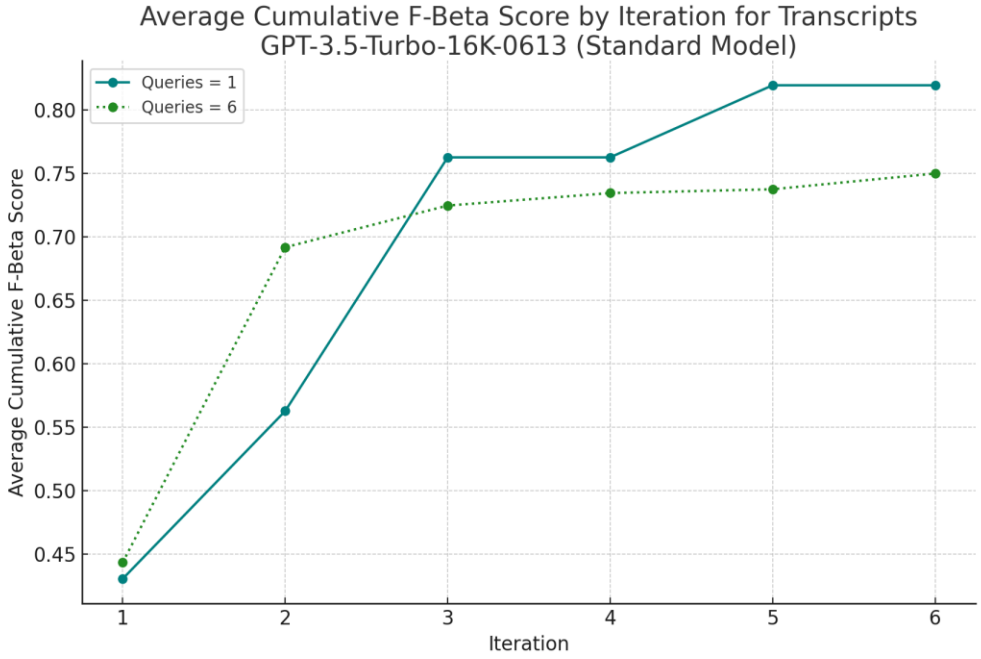


Figure 8. Average Cumulative F-Beta Score by Iteration for Police Reports GPT-3.5-Turbo-4K-0613 (Standard Model)

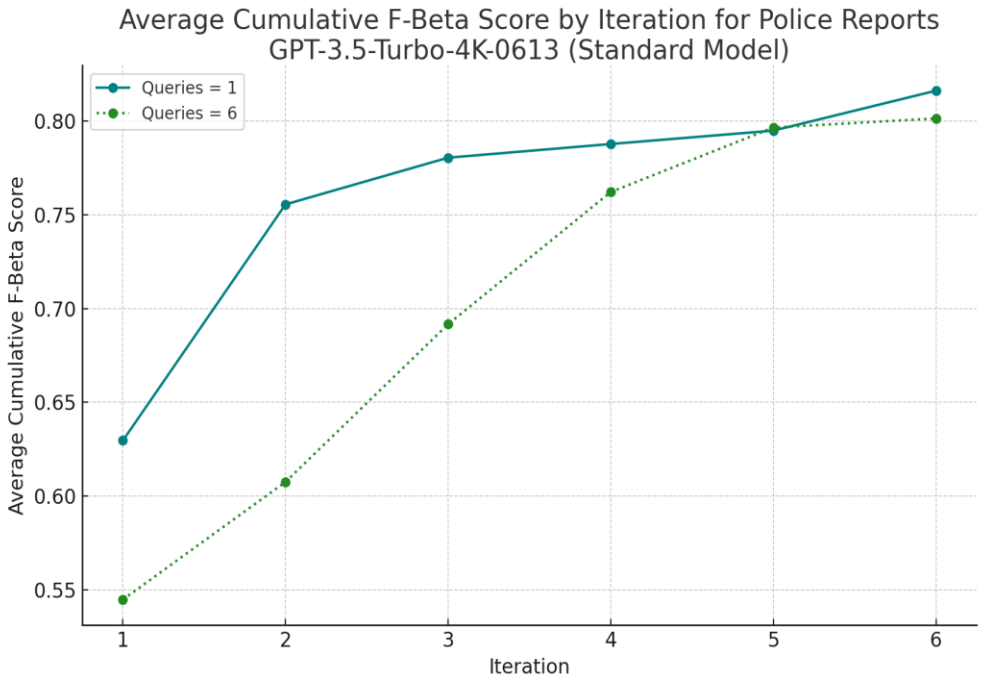
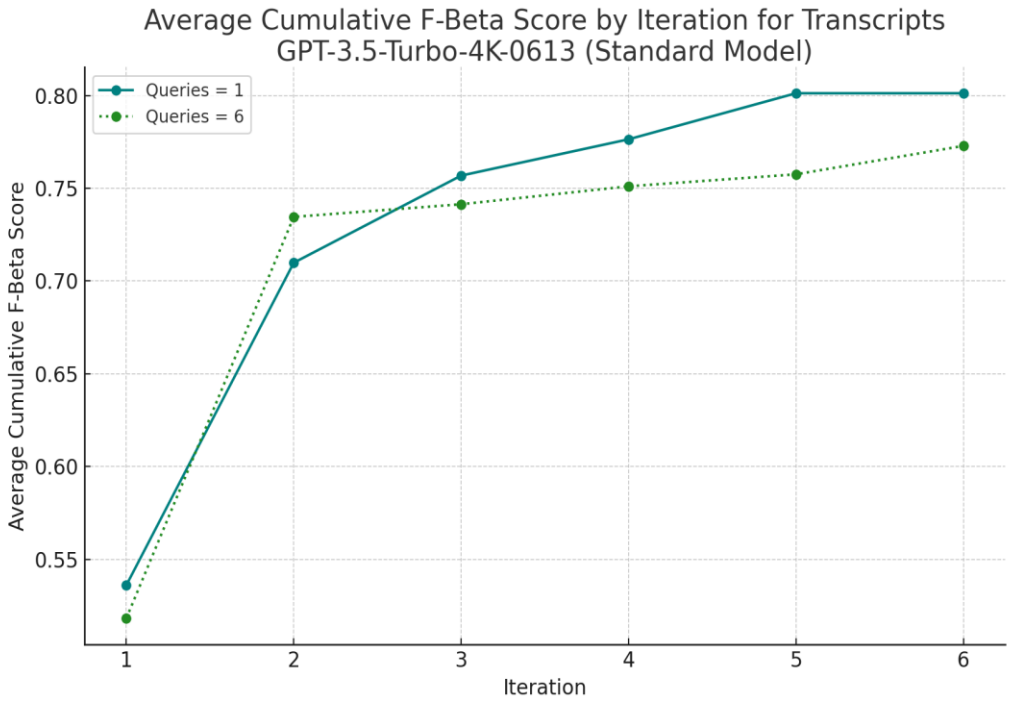


Figure 9. Average Cumulative F-Beta Score by Iteration for Transcripts GPT-3.5-Turbo-4K-0613 (Standard Model)



In evaluating the model's iterative performance, we observed that the point of diminishing returns generally occurs after the fourth iteration. While subsequent iterations do yield improvements, the incremental benefits should be weighed against computational and economic costs. Our comparative analysis using six unique queries versus one comprehensive query revealed that neither approach consistently outperformed the other. Instead, their effectiveness varied depending on the document type and the specific test.

The cumulative F-beta scores from the final iteration indicate that each model's proficiency varied based on the testing conditions. The GPT-4-8K model excelled in handling transcripts, while the GPT-3.5-Turbo-16K model demonstrated a slight advantage in processing reports. On the other hand, the GPT-3.5-Turbo-4K model showcased a balanced performance across different document types, with notable improvement over both of the more advanced models when employing a single query in reports. These findings underscore the importance of a strategic approach to model selection, as specific models may yield better results with particular types of documents or query strategies.

IV Fine-Tuning the Large Language Model

At the time of our tests, the cost differences between GPT-4-0613 and GPT-3.5-0613 were significant, with GPT-4's cost per input and output token being 1900% and 2900% higher, respectively. Given these substantial economic factors, our next iteration of model evaluation considered fine-tuning the GPT-3.5-Turbo-4K-0613, which was then the only model available for this purpose.

Fine-tuning involves tailoring a pre-trained model to better suit specific tasks or datasets[10]. By adjusting the model's parameters through additional training on a targeted dataset, fine-tuning aims to enhance the model's performance for particular applications[11]. In our case, the objective of fine-tuning GPT-3.5-Turbo-4K-0613 was to approximate the advanced performance of GPT-4 and GPT-3.5-Turbo-16k-0613, leveraging GPT-3.5-Turbo-4K-0613's existing capabilities while optimizing it for the specific nuances and complexities found in exoneration documents.

To train the GPT-3.5-Turbo-4K-0613 for handling exoneration documents, we generated training data using GPT-4-0613. GPT-4-0613, with its advanced capabilities, produced outputs that replicate the issues found in actual exoneration documents, such as inconsistent OCR quality, typos, and complex syntactic structures. These outputs were then utilized to train GPT-3.5-Turbo-4K to be more adept at analyzing exoneration documents.

Listing 5.0: **JSON Training Data Example for Fine-Tuning GPT-3.5-Turbo-4K-0613**

{
"messages":
[{ "role": "system", "content":

- "As an AI assistant, my role is to meticulously analyze criminal justice documents and extract information about law enforcement personnel. The response will contain: 1) The name of a law enforcement personnel. The individual's name must be prefixed with one of the following titles to be in law enforcement: Detective, Sergeant, Lieutenant, Captain, Deputy, Officer, Patrol Officer, Criminalist, Technician, Coroner, or Dr. Please prefix the name with 'Officer Name: I will derive this data from the following paragraph: On September 13, DET. X. Y. Allen responded to claims of counterfeit money circulating in the Westside Market. Primary informants were Mrs. Jacobs, a vendor, and Mr. Silva, a customer. FORWARD COPY TO: DETECTIVE DIVISION. INFORMANT DOCS."},

{ "role": "user", "content":

- 'Identify each individual in the transcript, by name, who are directly referred to as officers, sergeants, lieutenants, captains, detectives, homicide officers, and crime lab personnel." },

{ "role": "assistant", "content":"Officer Name: DET. X. Y. Allen

- Officer Context: On September 13, DET. X. Y. Allen responded to claims of counterfeit money circulating in the Westside Market.
- Officer Role: Detective" }]

}

To assess the impact of fine-tuning on the GPT-3.5-Turbo-4K-0613 model's performance, we established a set of benchmarks using three baseline models: the Standard GPT-4-8K-0613, the Standard GPT-3.5-Turbo-16K-0613, and the Standard GPT-3.5-Turbo-4K-0613. These models provided a comparative framework from which we could measure the incremental performance gains achieved through fine-tuning efforts on datasets of 25 to 300 labels.

[10] Evani Radiya-Dixit & Xin Wang, "How fine can fine-tuning be? Learning efficient language models" (2024) ArXiv: 2004.14129, online: <arxiv.org/abs/2004.14129>.

[11] Alexander Dunn *et al*, "Structured information extraction from complex scientific text with fine-tuned large language models" (2022) ArXiv:2212.05238, online: <arxiv.org/abs/2212.05238>.

Initially, the results of fine-tuning were mixed; the GPT-3.5-Turbo-4K-0613 models trained on smaller datasets often performed worse than the baseline models, including the standard GPT-3.5-Turbo-4K model. However, once fine-tuned with 300 labels, the performance of the GPT-3.5-Turbo-4K-0613 model showed significant advancement. For six queries in police reports, it attained a score of 0.859, surpassing the previous high baseline score of 0.842. In transcripts with one query, the fine-tuned model achieved a score of 0.849, outperforming all baseline models, which had a high of 0.819. Across all other test conditions, the fine-tuned model performed comparably to the baseline models, clearly demonstrating that fine-tuning can significantly enhance model performance beyond the established standards for analyzing exoneration documents.

As we continue to explore the potential of large language models, understanding their performance variability—especially under different conditions and datasets—is important. Our current comparative analysis selects the highest scores from five iterations for each standard and fine-tuned model variant. Moving forward, we plan to conduct a more rigorous investigation involving 100 individual runs per model. This approach will provide a robust dataset that captures the full scope of each model's capabilities. We will record and analyze the performance data for each run, focusing on police reports and transcripts, with one and six queries, to accurately assess the models' stability and reliability. Statistical measures such as mean performance scores, confidence intervals, and standard deviations will be employed to quantify the models' consistency. The data obtained will be instrumental in enhancing model performance and fine-tuning application strategies, ensuring that the models deliver consistent and reliable results.

Figure 10. Comparative Performance Analysis of GPT-3.5-Turbo Variants on Reports and Transcripts

Model Variant	Data Type	Performance with 1 Unique Query	Performance with 6 Unique Queries
GPT-4-8k-0613 (Standard)	Reports	0.732	0.773
GPT-4-8k-0613 (Standard)	Transcripts	0.807	0.821
GPT-3.5-Turbo-16k-0613 (Standard)	Reports	0.807	0.842
GPT-3.5-Turbo-16k-0613 (Standard)	Transcripts	0.819	0.749
GPT-3.5-Turbo-4k-0613 (Standard)	Reports	0.816	0.801
GPT-3.5-Turbo-4k-0613 (Standard)	Transcripts	0.801	0.772
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 25 labels)	Reports	0.771	0.827

GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 25 labels)	Transcripts	0.737	0.815
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 50 labels)	Reports	0.821	0.8
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 50 labels)	Transcripts	0.793	0.759
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 100 labels)	Reports	0.794	0.803
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 100 labels)	Transcripts	0.727	0.802
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 200 labels)	Reports	0.816	0.793
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 200 labels)	Transcripts	0.815	0.797
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 300 labels)	Reports	0.819	0.859
GPT-3.5-Turbo-4k-0613 (Fine-Tuned with 300 labels)	Transcripts	0.849	0.771

V Entity Resolution and Entity Matching

After extracting structured information about individuals involved in exonerations from the case documents, we will have a database of *raw mentions*. A mention consists of a name, the natural language context of the mention, a role, and various pieces of case-level metadata, including case number, date, and jurisdiction.

The same entity can be mentioned repeatedly, within a single document, across many documents related to a single case, and even across cases. Due to natural variations in how names are reported, including the varying contexts in which they appear and even OCR noise, mentions that refer to the same entity – "co-referent" mentions – will look different. For example: "Detective Tom Jones" in one mention may be referred to as "Det. Jones" or even just "Jones" in other mentions. Our goal is to match every mention to a canonical reference record in LLEAD. However, some mentions in isolation do not provide sufficient information to match the mention directly to LLEAD. A mention such as "Detective Jones" could correspond to multiple candidate records. By clustering co-referent mentions, we can aggregate information about distinct entities to give ourselves the best chance at a successful match. For example, knowing that the mention of "Detective Jones" refers to the same person as a previous mention of "Detective Tom Jones of the New Orleans PD" provides us with the context necessary to match *both* mentions to the correct reference record.

Entity resolution, or deduplication, and entity matching both rely on an appropriate pairwise similarity measure – for two given mentions, or for one mention and a reference record in the database, we want to be able to measure how "similar" they are, where we are defining "similarity" in terms of how likely they are to co-refer to the same real-life entity. Using this metric, we then want to cluster mentions that are similar to each other so that we can treat mentions that get clustered together as co-referent. By combining information within clusters, we can build an aggregate mention record that is as complete as possible. Finally, we can match the aggregate mention to our canonical database, once again using a pairwise similarity metric.

The entity resolution process is described in detail in Peter Christen's *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection (Data-Centric Systems and Applications)*. Traditionally, the inputs to this process are structured records. A novel challenge in our work is that a significant portion of each mention, the "context", consists of unstructured text. String similarity, using metrics such as Jaccard or edit distance, may be sufficient to capture similarity in names, but in order to compare two mention contexts, we need to capture semantic similarity. By *embedding* – or, mapping the text to vectors of real numbers – the mentions using natural language models, we can translate the mentions into a format that allows us to measure semantic similarity using vector databases that enable efficient nearest-neighbor searching. Embedding mentions for the purposes of entity normalization are described in Learning Text Similarity with Siamese Recurrent Networks[12] and NSEEN: Neural Semantic Embedding for Entity Normalization[13]. As those examples demonstrate, we seek an embedding that captures both the syntactic similarity of names across mentions as well as the semantic similarity of contexts. Given the appropriate embeddings, we can compare mentions using cosine similarity. As those readings also describe how to learn appropriate embeddings without much training data, we propose once again using embeddings from large language models to embed the contexts, further reducing our need for hand-labeled data.

At this point in the deduplication process, scale becomes a concern. Even with a good pairwise similarity metric between mentions, given the number of distinct mentions that we are extracting, we cannot realistically calculate similarity for every combination of two extracted mentions. However, such a pairwise *similarity matrix* is usually a precondition for clustering. A common solution is blocking[14]. This involves identifying "candidate" pairs that could be co-referent, while filtering out sufficiently dissimilar pairs without ever considering them. One approach to blocking that captures our notion of syntactic similarity in names is to only consider pairs that match on some substring. Representing our mentions as embeddings allows us to unlock another family of powerful candidate selection techniques, known as *locality sensitive hashing* (LSH). LSH describes a family of techniques that are based on mapping input mentions to buckets in such a way that similar mentions get mapped to the same bucket. These methods are described

[12] Paul Neculoiu, Maarten Versteegh & M Rotaru, "Learning Text Similarity with Siamese Recurrent Networks" (2016) Proc 1st Workshop on Representation Learning for NLP 148, online (pdf): <aclanthology.org/W16-1617.pdf>.

[13] Shobeir Fakhraei, Joel Mathew & Jose Luis Ambite, "NSEEN: Neural Semantic Embedding for Entity Normalization" (2018) ArXiv:1811.07514, online: <arxiv.org/abs/1811.07514>.

[14] Patrick Ball, "How do we find duplicates among multiple, giant datasets?" (last accessed 5 Dec 2023) online (blog): <hrdag.org/tech-notes/adaptive-blocking-writeup-1.html>.

in *Mining of Massive Datasets*[15]. In some instances, we may even be able to skip a second clustering step altogether, treating the LSH buckets as our final clusters.

Assuming we still need a separate clustering step, we have a range of unsupervised and semi-supervised clustering algorithms we can make use of. However, as described in "Theoretical Limits of Record Linkage and Microclustering"[16] and "Flexible Models for Microclustering with Application to Entity Resolution,"[17] the data we want to cluster is different from the data assumed by most clustering models in ways that affect the quality of solutions that those models can provide. Specifically most generative models for clustering implicitly assume that the number of data points in each cluster grows linearly with the total number of data points… However, for some applications, this assumption is inappropriate. For example, when performing entity resolution, the size of each cluster should be unrelated to the size of the data set, and each cluster should contain a negligible fraction of the total number of data points."[18]

Ball introduces hierarchical agglomerative clustering as a robust solution to the specific problem of clustering for entity resolution.[19] NSEEN, on the other hand, relies on the "sketch" technique described in Mining of Massive Datasets for a type of locality sensitive hashing that approximates clusters based on cosine distance in the embedding space.[20] In addition to these techniques, we will try to build on our success using HyDE and large language models to improve the matching and clustering steps.

Once we have co-referent clusters, we will attempt to use the same pairwise similarity metric, as well as a new one specific to the LLEAD data, in order to match resolved entities to canonical records in the reference database.

VI Future Research

Future research will concentrate on fine-tuning and optimizing the latest developments from OpenAI, particularly the GPT-3.5-Turbo-16K-1106 and GPT-4-Turbo models. Our experiments will focus on running similar tests, such as testing the effects of chunk size, chunk overlap, and 'k' value, in addition to fine-tuning, as the new GPT-3.5-Turbo-16K-1106 model is available for fine-tuning.

---

[15] Jure Leskovec, Anand Rajaraman & Jeff Ullman, "Mining of Massive Datasets" (last accessed 5 Dec 2023), online: <mmds.org>.

[16] James E Johndrow, Kristian Lum & David B Dunson, "Theoretical Limits of Record Linkage and Microclustering" (2017) ArXiv:1703.04955, online: <arxiv.org/abs/1703.04955>.

[17] Giacomo Zanella *et al*, "Flexible Models for Microclustering with Application to Entity Resolution" (2016) ArXiv:1610:09780, online: <arxiv.org/abs/1610.09780> at 1.

[18] Giacomo Zanella, Brenda Betancourt, Hanna Wallach, Jeffrey Miller, Abbas Zaidi, Rebecca C. Steorts, "Flexible Models for Microclustering with Application to Entity Resolution," [online] available: <https://www.microsoft.com/en-us/research/uploads/prod/2016/11/Flexible-Models-for-Microclustering-with-Application-to-Entity-Resolution.pdf> (accessed 12/5/2023).

[19] Patrick Ball, "Clustering and solving the right problem" (25 Jul 2016) online (blog): <hrdag.org/tech-notes/clustering-and-solving-the-right-problem.html>

[20] Shobeir Fakhraei, Joel Mathew & Jose Luis Ambite, "NSEEN: Neural Semantic Embedding for Entity Normalization" (2018) ArXiv:1811.07514, online: <arxiv.org/abs/1811.07514>.

While our current fine-tuned GPT-3.5-Turbo-4K-0613 model stands as the most efficient iteration so far, it's important to acknowledge that the new GPT-4-Turbo outperforms this model. In its baseline setup with a chunk size of 500, an overlap of 250, and a 'k' value of 20, the GPT-4-Turbo has achieved F-beta scores of up to 0.85. Further adjustments of its parameters to a chunk size of 20000, an overlap of 6000, and a 'k' value of 50 have led to F-Beta scores reaching as high as 0.95. This impressive performance highlights the potential for substantial improvements in structured data extraction tasks, offering a promising direction for future research and model optimization.