# COVID-19 Detection from Unstructured Medical Text: Project Report

## Executive Summary

This report documents the development and implementation of a comprehensive system for detecting COVID-19 from unstructured medical text. The project addresses the critical challenge faced by healthcare professionals during the COVID-19 pandemic: distinguishing COVID-19 from other respiratory conditions based on clinical notes and symptom descriptions before test results are available.

We implemented a two-stage pipeline that combines Named Entity Recognition (NER) with BioBERT-based classification. The system extracts relevant medical entities from clinical text and uses them, alongside structured patient data, to predict COVID-19 likelihood. Our model achieved 92.8% accuracy and an ROC AUC of 0.967 on test data, significantly outperforming traditional machine learning approaches.

Additionally, we developed "Harvey," a chatbot interface that makes this technology accessible to healthcare professionals through natural language interaction. The interface allows clinicians to explore patient data, analyze symptoms, and receive COVID-19 risk assessments based on our AI pipeline.

This project demonstrates the potential of combining NLP techniques with domain-specific transformer models to extract valuable diagnostic information from unstructured medical text, with implications beyond COVID-19 for medical text analysis more broadly.

## 1. Introduction

### 1.1 Problem Statement

For our project, we wanted to try and use natural language processing to predict if patients have COVID-19. This report will take you through a comprehensive approach to detecting COVID-19 from unstructured medical text using artificial intelligence techniques.

One powerful application of this approach emerged during the early stages of the pandemic. In early 2020, users on Twitter, Reddit, and other forums began posting about symptoms such as fever, cough, and loss of smell—often before official case reports showed any increase. These social media posts served as unfiltered, real-time signals of disease activity.

Natural Language Processing (NLP) models were able to scan millions of public posts for relevant keywords and symptom clusters. For example, spikes in mentions of "loss of smell" in New York City were detected up to 10 days before a confirmed surge in COVID-19 cases. These insights were critical for public health agencies, who used them to identify emerging hotspots, allocate testing resources, and implement early interventions. This highlighted the significant value of unstructured data in epidemic intelligence and the proactive capabilities of AI-powered surveillance.

Early in the COVID-19 pandemic, healthcare professionals faced a significant challenge: many respiratory illnesses share similar symptoms, making it difficult to distinguish COVID-19 from conditions like seasonal flu, common cold, or allergies without waiting for test results. With limited testing capacity and delays in results, clinicians needed better tools to help them make preliminary assessments based on available information - primarily clinical notes and patient-reported symptoms.

The key question our project addresses is: Can advanced NLP techniques effectively analyze unstructured medical text to identify likely COVID-19 cases and support clinical decision-making?

### 1.2 Project Goals

Our project aimed to achieve the following objectives:

1. Develop a robust NER system for extracting medical entities from unstructured clinical text
2. Implement a classification pipeline using BioBERT to predict COVID-19 likelihood
3. Evaluate performance against traditional machine learning approaches
4. Create a practical interface (Harvey chatbot) for healthcare professionals
5. Demonstrate a generalizable approach that could be adapted to other medical conditions

### 1.3 Technical Background

This project sits at the intersection of several technical domains:

**Named Entity Recognition (NER)** is a subtask of information extraction that seeks to locate and classify named entities in text into predefined categories. In the medical domain, NER can identify entities like symptoms, diseases, medications, and temporal expressions.

**Transformer models** represent a breakthrough in natural language processing, using self-attention mechanisms to process text. BioBERT is a domain-specific transformer pre-trained on biomedical literature, making it particularly suitable for understanding medical terminology and contexts.

**Clinical NLP** applies natural language processing techniques to clinical text, which presents unique challenges including specialized vocabulary, abbreviations, and telegraphic documentation styles.

## 2. How Data Flows Through Our System

Our COVID-19 detection system processes data in a straightforward pipeline that's easy to understand. Here's how information moves through our project in simple terms:

### Step 1: Data Collection

First, we gather medical notes from several sources: - Clinical notes from doctors describing patient symptoms - Medical records with patient information - CDC COVID-19 case data for patterns and training

### Step 2: Finding Important Information in the Text

When a doctor writes a note like "Patient has fever, dry cough for 3 days, and reports loss of taste," our system: 1. Reads through the text 2. Picks out important words and phrases (fever, cough, loss of taste) 3. Identifies how long symptoms have been present (3 days) 4. Notes how severe symptoms are (mild, moderate, severe)

This is like highlighting the most important parts of the medical notes.

### Step 3: Converting Text to Numbers

Computers need numbers to work with, so we convert the highlighted information into data the computer can understand: - Create yes/no flags for key COVID symptoms (Has fever? Yes = 1, No = 0) - Count the number of symptoms mentioned - Record when symptoms started - Note severity levels

### Step 4: Combining Information

We merge the information extracted from the notes with other patient data like: - Age and gender - Existing health conditions - Lab test results (if available)

This gives us a complete picture of each patient case.

### Step 5: Making Predictions

Our system uses two approaches to predict if the patient might have COVID-19: 1. **BioBERT Model** - An AI system trained on medical text that understands context and medical terminology 2. **Traditional Machine Learning** - Analyzes patterns in the symptom data

### Step 6: Presenting Results

Finally, the system shows the results in an easy-to-understand format: - COVID-19 risk score (percentage) - Highlighted symptoms that influenced the prediction - Confidence level in the prediction

Healthcare professionals can interact with this information through our Harvey chatbot, asking questions like "What symptoms does this patient have?" or "What's the COVID risk for this patient?"

### The Advantage of Our Approach

Unlike systems that only use laboratory test results or structured data, our approach: - Works when test results aren't yet available - Makes use of detailed information in doctors' notes - Understands medical terminology and context - Provides explainable results (shows which symptoms led to the prediction) - Can be accessed through natural conversation with the Harvey chatbot

This helps doctors make faster decisions about patient care, isolation needs, and test prioritization when resources are limited.

## 3. Technical Approach

### 3.1 Pipeline Overview

Our system implements a two-stage pipeline:

1. **Named Entity Recognition (NER)**: Extracts relevant medical entities from unstructured clinical text
2. **Classification**: Uses extracted entities and structured patient data to predict COVID-19 likelihood

The pipeline processes raw clinical notes through multiple stages: - Text preprocessing and normalization - Entity extraction via NER - Feature engineering from extracted entities - Integration with structured patient data - BioBERT-based classification - Probability calibration and output

### 3.2 Data Sources

We utilized multiple data sources throughout this project:

1. **MIMIC-IV Clinical Dataset**: De-identified electronic health records containing clinical notes, patient demographics, diagnoses, lab results, and outcomes. This dataset provided realistic medical text with the complexity we'd expect in real-world applications.
2. **CDC COVID-19 Case Surveillance Data**: Structured data on COVID-19 cases including demographics, symptoms, comorbidities, and outcomes. This helped us understand typical symptom patterns and prevalence.
3. **CORD-19 Research Corpus**: A resource of scientific papers about COVID-19 and related coronaviruses. This provided valuable context from scientific literature about COVID-19 symptoms and progression.

We created integrated datasets by combining these sources, with appropriate preprocessing and de-identification steps to maintain privacy compliance.

### 3.3 Named Entity Recognition Implementation

We implemented three different NER approaches, each with distinct advantages:

#### 3.3.1 Rule-Based NER

The rule-based approach uses regular expressions and keyword matching to identify entities in text. While simple, this approach is fast, interpretable, and doesn't require training data.

Key implementation details: - Pattern matching for COVID-19 symptoms from a predefined list - Regular expressions for capturing time expressions (e.g., "3 days ago") - Pattern matching for severity indicators (e.g., "mild," "severe")

Example code snippet from our implementation:

def \_compile\_patterns(self, terms, pattern\_template):  
 """Compile regex patterns for a list of terms."""  
 patterns = []  
 for term in terms:  
 # Escape special regex characters in the term  
 escaped\_term = re.escape(term)  
 # Format the pattern template with the escaped term  
 pattern = pattern\_template.format(escaped\_term)  
 # Compile the regex pattern  
 patterns.append(re.compile(pattern, re.IGNORECASE))  
 return patterns

#### 3.3.2 spaCy-Based NER

The spaCy-based approach leverages the spaCy NLP library for more sophisticated NER. This approach allows custom training and more context-aware entity recognition.

Key implementation details: - Custom training capability using annotated medical text - Integration with spaCy's linguistic features - Support for custom entity types relevant to COVID-19

#### 3.3.3 Transformer-Based NER

The transformer-based approach uses biomedical pre-trained models for state-of-the-art performance. This approach is the most powerful but also the most computationally intensive.

Key implementation details: - Integration with Hugging Face transformers library - Support for biomedical pre-trained models like BioBERT - Token-level classification for precise entity boundaries

### 3.4 Feature Engineering

After extracting entities using NER, we transform them into structured features suitable for classification:

1. **Entity-based features**:
2. Symptom presence (binary flags for each COVID-related symptom)
3. Symptom counts (total number of symptoms mentioned)
4. Time-related features (recent onset, duration of symptoms)
5. Severity indicators (mild, moderate, severe)
6. **Integrated features**:
7. Demographic information (age, gender)
8. Risk factors and comorbidities
9. Lab values when available
10. Exposure history

The feature engineering process ensures that the rich information contained in unstructured text is converted into a format that machine learning models can effectively use.

### 3.5 BioBERT Classification Model

Our classification approach leverages BioBERT, a domain-specific variant of BERT pre-trained on biomedical literature.

#### 3.5.1 Model Architecture

We used the dmis-lab/biobert-base-cased-v1.1 model as our foundation, which has: - 12 transformer layers - 768 hidden dimensions per token - 12 attention heads - Approximately 110 million parameters - Pre-training on PubMed abstracts and PMC full-text articles

#### 3.5.2 Training Methodology

The model was fine-tuned on our integrated dataset with the following configuration: - Optimizer: AdamW - Learning rate: 2e-5 with linear decay - Batch size: 16 - Epochs: 3 - Training strategy: Fine-tuning all layers

#### 3.5.3 Input Processing

Clinical notes undergo a specific processing pipeline before being fed to the BioBERT model: 1. NER extraction of medical entities 2. Formatting extracted entities into structured representation 3. Tokenization using BioBERT tokenizer 4. Truncation/padding to maximum sequence length (512 tokens) 5. Addition of special tokens ([CLS] for classification, [SEP] for separation)

## 4. Results and Evaluation

### 4.1 Performance Metrics

Our BioBERT model achieved the following performance metrics on the test set:

| Metric | Value |
| --- | --- |
| Accuracy | 92.8% |
| Precision | 0.94 |
| Recall | 0.91 |
| F1 Score | 0.925 |
| ROC AUC | 0.967 |
| PR AUC | 0.963 |

These results indicate strong performance in COVID-19 detection from clinical text, with a good balance between precision and recall.

### 4.2 Comparative Performance

We compared our BioBERT approach with traditional machine learning models trained on the same features:

| Model | ROC AUC | F1 Score |
| --- | --- | --- |
| BioBERT | 0.967 | 0.925 |
| Logistic Regression | 0.843 | 0.783 |
| Random Forest | 0.921 | 0.867 |
| Gradient Boosting | 0.937 | 0.884 |

While gradient boosting performed reasonably well, BioBERT consistently outperformed all traditional approaches, demonstrating the value of transformer-based models for this task.

### 4.3 Error Analysis

We conducted a detailed error analysis to understand when and why our model fails:

**False Negatives**: Most common in cases with atypical symptom presentation, where patients had COVID-19 but presented with unusual or very mild symptoms. This suggests that our model relies heavily on the presence of "typical" COVID symptoms.

**False Positives**: Most frequent in cases with similar symptoms to COVID-19, particularly influenza cases with fever and respiratory symptoms. This highlights the inherent challenge in distinguishing these conditions based on symptoms alone.

**Edge Cases**: The model struggled with asymptomatic COVID-19 cases and patients with multiple comorbidities that complicated the clinical picture.

### 4.4 NER Performance

We also evaluated the performance of our different NER approaches:

| NER Approach | Precision | Recall | F1 Score |
| --- | --- | --- | --- |
| Rule-based | 0.85 | 0.76 | 0.80 |
| spaCy-based | 0.89 | 0.82 | 0.85 |
| Transformer | 0.93 | 0.90 | 0.91 |

The transformer-based NER performed best, particularly for complex or ambiguous symptoms, but at the cost of higher computational requirements.

## 5. Harvey Chatbot Implementation

### 5.1 Overview and Architecture

Harvey is a chatbot interface designed for healthcare professionals to interact with our COVID-19 detection system. The architecture consists of:

* **Frontend**: Clean, minimalist HTML/CSS/JavaScript interface
* **Backend**: Flask-based server implementing the NER+BioBERT pipeline
* **Data Layer**: JSON-based patient records with clinical notes

### 5.2 Key Features

Harvey implements several key features:

1. **Patient Selection**: Browse and select patient records from a sidebar
2. **Clinical Analysis**: View symptoms, lab results, and COVID-19 probability
3. **Natural Language Interface**: Ask questions about patients in natural language
4. **Real-time Analysis**: Process clinical notes to extract relevant entities
5. **Visualization**: Display of extracted entities from clinical text
6. **Suggestion Bullets**: Clickable suggestions for common clinical questions

### 5.3 Technical Implementation

The chatbot implementation includes several notable technical aspects:

**Multi-level Fallback System**: The system implements both server-side and client-side fallbacks to ensure the chatbot always responds, even when the backend encounters issues:

@app.route('/api/chat', methods=['POST'])  
def chat():  
 try:  
 data = request.get\_json()  
 user\_message = data.get('message', '')  
 patient\_id = data.get('patient\_id', None)  
  
 # Try to analyze the user's message and generate a response  
 try:  
 response = analyze\_patient\_text(user\_message, patient\_id)  
 except Exception as analysis\_error:  
 print(f"Error in patient text analysis: {analysis\_error}")  
 # Fallback responses based on keywords in the user's message  
 response = get\_fallback\_response(user\_message, patient\_id)  
  
 return jsonify({  
 "success": True,   
 "response": response,  
 "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 })  
 except Exception as e:  
 # Final fallback if everything else fails  
 return jsonify({  
 "success": True,  
 "response": "I'm having trouble processing that request...",  
 "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")  
 })

**Responsive Design**: The interface adapts to different screen sizes while maintaining usability:

@media (max-width: 768px) {  
 .container {  
 flex-direction: column;  
 }  
  
 .sidebar {  
 width: 100%;  
 height: auto;  
 max-height: 40vh;  
 }  
  
 .main-content {  
 height: 60vh;  
 }  
}

**Interactive Elements**: The interface includes interactive elements like suggestion bullets that enhance the user experience:

function addSuggestionBullets(patientId) {  
 const suggestionContainer = document.createElement('div');  
 suggestionContainer.className = 'quick-suggestions';  
  
 const title = document.createElement('div');  
 title.className = 'suggestion-title';  
 title.textContent = 'Suggested questions:';  
 suggestionContainer.appendChild(title);  
  
 const bullets = document.createElement('div');  
 bullets.className = 'suggestion-bullets';  
  
 const suggestions = [  
 "What are the patient's main symptoms?",  
 "When did symptoms first appear?",  
 "What is the COVID-19 probability?",  
 "Are there any abnormal lab results?",  
 "What risk factors does this patient have?"  
 ];  
  
 suggestions.forEach(suggestion => {  
 const bullet = document.createElement('div');  
 bullet.className = 'suggestion-bullet';  
 bullet.textContent = suggestion;  
 bullet.addEventListener('click', () => {  
 document.getElementById('message-input').value = suggestion;  
 document.getElementById('send-button').click();  
 });  
 bullets.appendChild(bullet);  
 });  
  
 suggestionContainer.appendChild(bullets);  
 return suggestionContainer;  
}

## 6. Practical Applications and Impact

### 6.1 Clinical Use Cases

Our system addresses several important clinical use cases:

1. **Triage Support**: Helping clinicians prioritize which patients need immediate attention or testing when resources are limited
2. **Early Detection**: Identifying likely COVID-19 cases before test results are available
3. **Documentation Assistance**: Extracting structured information from clinical notes
4. **Research Support**: Analyzing large amounts of clinical text for patterns

### 6.2 Deployment Considerations

For deployment in clinical settings, several considerations are important:

1. **Computational Requirements**: The full pipeline with transformer-based NER and BioBERT classification requires significant computational resources. For resource-constrained environments, the rule-based NER with gradient boosting classification provides a lighter alternative with reasonable performance.
2. **Privacy and Security**: Working with medical data requires strict privacy controls and compliance with regulations like HIPAA. Our implementation ensures all patient data is de-identified and encrypted during transmission.
3. **Model Interpretability**: Healthcare professionals need to understand why a model makes a prediction. Our system provides feature importance information to explain which symptoms most influenced the COVID-19 probability score.
4. **Integration with EHR Systems**: For practical use, the system should integrate with existing Electronic Health Record systems. Our API-based design facilitates this integration.

## 7. Technical Challenges and Solutions

Throughout the project, we encountered several technical challenges:

### 7.1 Working with Clinical Text

Clinical text presents unique challenges due to its specialized vocabulary, abbreviations, and telegraphic style. We addressed these through:

* Domain-specific preprocessing steps
* Medical abbreviation expansion
* Specialized tokenization for clinical text
* Use of biomedical pre-trained models

### 7.2 Data Imbalance

COVID-19 positive cases were underrepresented in our training data. We addressed this through:

* Stratified sampling for train/test splits
* Class weighting during model training
* Data augmentation for minority class
* Evaluation metrics appropriate for imbalanced data (ROC AUC, PR AUC)

### 7.3 Entity Extraction Challenges

Extracting medical entities from text proved challenging due to the variety of ways symptoms are described. We implemented multiple NER approaches to address this, combining their strengths:

* Rule-based NER for common, well-defined symptoms
* spaCy NER for context-aware extraction
* Transformer-based NER for complex cases

## 8. Future Directions

The current project demonstrates the potential of NLP for COVID-19 detection, but several promising directions for future work remain:

### 8.1 Multimodal Integration

Combining text analysis with other data types could enhance performance:

* Integration with medical imaging (chest X-rays, CT scans)
* Vital signs and continuous monitoring data
* Genomic data for variant identification

### 8.2 Expanded Clinical Applications

The techniques developed here could be adapted to other medical conditions:

* Other infectious diseases with overlapping symptoms
* Rare disease detection from clinical descriptions
* Mental health condition assessment from clinical notes

### 8.3 Technical Enhancements

Several technical improvements could further enhance the system:

1. **Temporal Modeling**: Incorporating symptom progression over time
2. **Multi-lingual Support**: Expanding beyond English to support global use
3. **Federated Learning**: Training models across institutions without sharing sensitive data
4. **Continuous Learning**: Updating models as new COVID variants emerge
5. **Explainable AI**: Enhancing model interpretability for clinical users

## 9. Conclusion

Our core problem focuses on detecting COVID-19 from unstructured clinical text. The motivation behind this project is to help clinicians and healthcare facilities triage patients more effectively and make quick, informed assessments—while also using limited resources more efficiently. The key question guiding our research is: Can advanced NLP techniques effectively analyze unstructured medical text to identify likely COVID-19 cases and support clinical decision-making? To answer this, we implemented a two-stage pipeline architecture and designed an interface to present results to stakeholders via a ChatBot. The ultimate goal is to automate the detection of COVID-19 from clinical notes, accelerating case identification and reducing the burden of manual review by tapping into the valuable, often overlooked data embedded in clinical documentation.

This project demonstrates the potential of combining NER with transformer models to extract valuable diagnostic information from unstructured medical text. Our COVID-19 detection system achieves high performance metrics and provides an accessible interface for healthcare professionals through the Harvey chatbot.

The approach developed here has broad implications beyond COVID-19, offering a template for how NLP can be applied to medical text analysis more generally. By converting unstructured clinical notes into structured, actionable information, such systems can help clinicians make more informed decisions and potentially improve patient outcomes.

As NLP techniques continue to advance, their integration into clinical workflows represents a significant opportunity to leverage the vast amounts of unstructured medical text generated daily in healthcare systems worldwide.

## Appendix A: Implementation Details

### A.1 Project Directory Structure

Disease\_Prediction\_Project/  
├── data/  
│ ├── raw/ # Raw data files  
│ ├── processed/ # Processed datasets  
│ ├── external/ # External datasets (MIMIC, CDC)  
│ │ └── mimic/  
│ │ └── note\_module/ # Clinical notes  
├── src/  
│ ├── data\_collection.py # Data gathering utilities  
│ ├── data\_processing.py # Data cleaning and preparation  
│ ├── ner\_extraction.py # NER implementations  
│ ├── data\_integration.py # Combines data sources  
│ ├── modeling.py # ML models and evaluation  
│ └── model\_evaluation.py # Performance metrics  
├── notebooks/  
│ ├── 01\_initial\_data\_exploration.ipynb  
│ ├── 02\_feature\_engineering.ipynb  
│ ├── 03\_model\_building.ipynb  
│ ├── 04\_ner\_extraction\_pipeline.ipynb  
│ └── master/  
│ └── Text\_Mining\_NER\_to\_BioBERT\_Pipeline.ipynb  
├── models/  
│ └── biobert\_covid\_classifier/ # Saved model files  
├── harvey\_chatbot/  
│ ├── app.py # Flask application  
│ ├── templates/ # HTML templates  
│ ├── static/ # CSS, JS, images  
│ └── utils/ # Utility functions  
├── tests/  
│ ├── test\_ner.py  
│ └── test\_mimic\_ner.py  
└── docs/  
 ├── project\_overview.md  
 └── biobert\_model.md

### A.2 Key Dependencies

The project relies on several key libraries:

* **Data Processing**: pandas, numpy
* **NLP**: spaCy, NLTK, regex
* **Machine Learning**: scikit-learn, PyTorch
* **Deep Learning**: transformers (Hugging Face)
* **Visualization**: matplotlib, seaborn
* **Web Interface**: Flask, JavaScript

### A.3 Model Hyperparameters

The final BioBERT model used the following hyperparameters:

training\_args = TrainingArguments(  
 output\_dir='./results',  
 num\_train\_epochs=3,  
 per\_device\_train\_batch\_size=16,  
 per\_device\_eval\_batch\_size=64,  
 warmup\_steps=0,  
 weight\_decay=0.01,  
 learning\_rate=2e-5,  
 logging\_dir='./logs',  
 logging\_steps=10,  
 evaluation\_strategy="epoch",  
 save\_strategy="epoch",  
 load\_best\_model\_at\_end=True,  
 metric\_for\_best\_model="f1"  
)

## Appendix B: Additional Results

### B.1 Feature Importance Analysis

Analysis of feature importance revealed the most predictive symptoms for COVID-19:

1. Loss of taste/smell (feature importance: 0.182)
2. Fever (feature importance: 0.143)
3. Dry cough (feature importance: 0.135)
4. Recent onset (< 1 week) (feature importance: 0.127)
5. Fatigue (feature importance: 0.118)

### B.2 Demographic Analysis

Performance varied slightly across demographic groups:

| Demographic Group | Accuracy | Recall | Precision |
| --- | --- | --- | --- |
| Age < 40 | 93.2% | 0.92 | 0.95 |
| Age 40-65 | 92.5% | 0.91 | 0.94 |
| Age > 65 | 91.8% | 0.89 | 0.93 |
| Male | 92.9% | 0.91 | 0.94 |
| Female | 92.6% | 0.90 | 0.95 |

These differences are relatively minor, suggesting the model performs consistently across demographics.

## Appendix C: Acknowledgments

We acknowledge the use of the following public resources:

* MIMIC-IV dataset, made available by the PhysioNet team
* CDC COVID-19 Case Surveillance Public Use Data
* CORD-19 dataset from the Allen Institute for AI
* Hugging Face transformers library
* BioBERT models from DMIS Lab