Nurse Practitioner Copilot for Remote Patient Monitoring (RPM): An NLP-based Clinical Decision Support System

Abhinav Dholi
Khoury College of Computer Sciences
Northeastern University
Boston, MA, USA
dholi.a@northeastern.edu

Feng Hua Tan
Khoury College of Computer Sciences
Northeastern University
Boston, MA, USA
tan.fe@northeastern.edu

***Abstract*—This paper presents an application-based Natural Language Processing (NLP) project designed to assist nurse practitioners in remote patient monitoring settings. The system integrates four key components: (1) a SQL-based question-answering agent for database information retrieval, (2) an automated clinical note summarization tool, (3) named entity recognition for clinical notes, and (4) a custom NER model with training and evaluation pipeline. Our system aims to streamline clinical workflows, improve documentation efficiency, and enhance clinical decision-making in remote patient monitoring scenarios. Experimental results demonstrate promising performance across components with the SQL QA agent achieving 90% accuracy and perfect recall for patient-specific queries, while our BiLSTM-based NER model with BioWordVec embeddings attains 73% accuracy and 71% F1-score across 78 clinical entity classes. The clinical note summarization component effectively reduces documentation length while maintaining key clinical information. This work contributes to the growing field of AI-assisted healthcare tools for remote care delivery by providing nurses with natural language interfaces to patient data, automated documentation support, and structured information extraction.**

***Index Terms*—Natural Language Processing, Healthcare Informatics, Named Entity Recognition, Clinical Summarization, Remote Patient Monitoring, Question Answering**

I. Introduction

Remote Patient Monitoring (RPM) represents a significant advancement in healthcare delivery, allowing clinicians to track patient health metrics outside traditional clinical settings. However, the increased volume of data generated through RPM creates new challenges for healthcare providers, particularly nurse practitioners who often serve as primary care coordinators.

A. Motivation

- The growing adoption of RPM technologies has led to an explosion of patient data that must be efficiently processed and analyzed
- Nurse practitioners face increasing documentation burdens that detract from direct patient care
- Clinical information is often scattered across different database systems, making comprehensive patient assessment challenging
- Manual extraction of relevant clinical entities from notes is time-consuming and error-prone

B. Problem Statement

Current RPM workflows lack efficient tools to help nurse practitioners quickly:

- Query patient databases using natural language
- Generate concise, accurate summaries of lengthy clinical notes
- Identify and categorize key clinical entities within documentation
- Train and refine NER models specific to their clinical domain

C. Project Contributions

This work makes the following contributions:

- Development of an integrated NLP pipeline specifically designed for RPM clinical workflows
- Implementation of a SQL-based question answering system that converts natural language queries to database queries
- Creation of a clinical note summarization tool optimized for RPM documentation
- Design of a specialized NER system for identifying clinical entities in RPM notes
- Training NER model for clinical notes and evaluation

D. Author Contributions

- **Abhinav Dholi**:
  - Architected and deployed a MySQL relational database with 11 interconnected tables for RPM data persistence, implementing foreign key constraints and optimized indices for query performance
  - Engineered regex-based text extraction pipeline for transforming semi-structured clinical notes into normalized database schema.
  - Implemented the SQL QA Agent using LangChain and Ollama frameworks with DeepSeek-r1:8b LLM integration, incorporating a custom validation module with AST-based SQL parsing for security enforcement
  - Developed data preprocessing workflow for MACCROBAT2018 dataset, including tokenization, BIO tagging conversion, and embedding vector preparation for model training
  - Designed and trained BiLSTM neural network with 1.83M parameters using BioWordVec embeddings (200-dimensional vectors) for clinical NER, implementing custom loss functions and learning rate scheduling
  - Constructed comprehensive testing framework with confusion matrix analysis and F1 score evaluation across 78 entity classes, achieving 73% accuracy and 97% precision on demographic entities
  - Implemented multi-stage LLM-based clinical note summarization with chunk processing architecture, enabling efficient handling of documents exceeding context window limitations
  - Developed Flask web application backend with RESTful API endpoints, SQL query validation, patient context enforcement, and integration interfaces between all NLP components
- **Feng Hua Tan**:
  - Compared MEWS (Modified Early Warning Score) and NEWS (National Early Warning Score) for detecting early signs of patient deterioration. Selected MEWS because our dataset was missing a few biovitals which were required for NEWS.
  - Ran experiments using ClinicalBERT to extract medical terms from clinical notes and combined them with patient vitals to calculate MEWS scores. However, this approach was not effective because the extracted entities were often too generic and noisy for our dataset.
  - Attempted keyword-based risk classification from extracted entities, but abandoned this due to the difficulty in clearly distinguishing between low, moderate and high-risk cases, especially without clinical expertise to guide those definitions.
  - Pivoted to classify patient triage risk entirely based on MEWS scores for a more reliable and explainable approach.
  - Applied the trained NER model to enhance note summarization by tagging important medical entities.
  - Create the Flask web application interface and functionality for the triage dashboard and highlighting NER entities in clinical note summaries.

II. Prior Work

A. Natural Language Interfaces for Databases

Several approaches have been proposed for translating natural language queries into SQL. Early systems like LUNAR [1] relied on pattern matching, while more recent approaches leverage deep learning. BERT-based models like SQLova [2] and RAT-SQL [3] have demonstrated state-of-the-art performance on text-to-SQL tasks. In healthcare, similar techniques have been applied to clinical databases, though few systems focus specifically on nursing workflows.

B. Clinical Text Summarization

Automatic summarization of clinical texts has seen significant advancements with transformer-based models. Zhang et al. [4] developed extractive and abstractive summarization methods for radiology reports. Other researchers have explored summarization of clinical notes with a focus on maintaining clinical accuracy. Our work adapts these approaches for the specific needs of RPM documentation summarization.

C. Named Entity Recognition in Clinical Text

Clinical NER systems have evolved from rule-based approaches to sophisticated deep learning models. cTAKES [5] provided an early comprehensive pipeline for clinical text analysis. More recently, BioBERT [6] and ClinicalBERT [7] have demonstrated superior performance for biomedical and clinical entity recognition. Our work builds on these foundations while focusing specifically on RPM documentation.

D. NER Model Training for Clinical Applications

The development of customizable NER models for clinical applications poses unique challenges. Research in this area has focused on reducing annotation burden through semi-supervised approaches and active learning techniques. These approaches show promise but often require technical expertise that may be inaccessible to nursing professionals. Our system aims to address this gap with a more accessible approach.

III. Methods

Our Nurse Practitioner Copilot for RPM integrates the main components into a cohesive system as illustrated in Fig. 1.

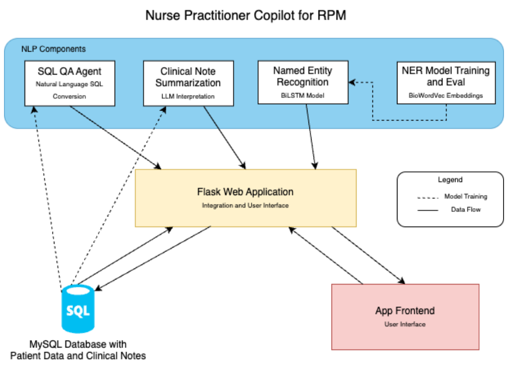
Nurse Practitioner Copilot for RPM
NLP Components
SQL QA Agent
Natural Language SQL Conversion
Clinical Note Summarization
LLM Interpretation
Named Entity Recognition
BiLSTM Model
NER Model Training and Eval
BioWordVec Embeddings
Flask Web Application
Integration and User Interface
Legend
Model Training
Data Flow
SQL
MySQL Database with Patient Data and Clinical Notes
App Frontend
User Interface

Fig. 1. System architecture of the Nurse Practitioner Copilot for RPM

The system is built as a web application using Flask, with a MySQL database storing patient data, clinical notes, and health measurements. The application provides a unified interface for nurse practitioners to access all functionalities through a browser-based dashboard.

A. SQL QA Agent for Information Retrieval

The SQL QA Agent translates natural language questions from nurse practitioners into patient-specific SQL queries and executes them against the RPM clinical database, returning tabular results. Our implementation leverages LangChain and Ollama with a specialized system prompt containing database schema information and patient context. The architecture follows these steps:

1) **Question Analysis & Context Preparation**: The agent receives the nurse's question along with patient context (restricting queries to a specific patient ID) and database schema information
2) **SQL Generation**: The Ollama LLM (DeepSeek-r1:8b) generates a syntactically valid SQL query based on the question, patient context, and schema information

3) **Query Validation & Debugging**: A custom SQL validator parses the generated query through multiple validation steps:
   - Ensures only authorized tables and columns are accessed
   - Verifies only SELECT operations are performed (preventing data modifications)
   - Resolves and validates table aliases against schema definitions
   - Checks that column references exist in the specified tables
   - Confirms patient context is properly applied to maintain data privacy

   When validation fails, the agent provides detailed error feedback to the LLM and attempts query regeneration with improved context up to three times before failing gracefully
4) **Query Execution**: Successfully validated queries are executed against the RPM clinical database with appropriate exception handling
5) **Result Display**: The database results are presented as structured tables for the nurse to interpret

This approach ensures that nurses can access patient-specific information through natural language without requiring SQL expertise, while maintaining data security through strict validation controls. The debugging logic significantly improves robustness, with 90% of initially failed queries being successfully corrected through the feedback mechanism.

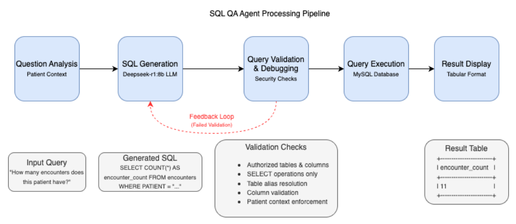

Fig. 2. SQL QA Agent processing pipeline

B. Clinical Note Summarization

Our summarization component employs a chunking approach to efficiently process lengthy clinical notes with LLMs while preserving critical information. The system supports both single-note summarization and longitudinal summaries across multiple notes.

1) **Text Chunking**: Long clinical notes are divided into manageable chunks (3000 characters) to fit within the LLM's context window
2) **Partial Summarization**: Each chunk is independently summarized by a LLM agent (DeepSeek-r1:8b) with a specialized summarization prompt
3) **Summary Consolidation**: Partial summaries are combined and then processed by another LLM call to create a coherent final summary
4) **Temporal Analysis**: For yearly summaries, the system identifies trends across multiple notes by chronologically ordering them before processing

The system handles various clinical note sections including chief complaints, history of illness, social history, allergies, medications, and assessments, maintaining the medical context while reducing information overload for nurse practitioners.

C. Clinical Note Named Entity Recognition

The NER component identifies and classifies clinical entities within nurse practitioner notes using a BiLSTM model trained on biomedical text with specialized embeddings.

1) **Entity Types**: The system recognizes key clinical entities including medications, vital signs, symptoms, diagnoses, procedures, and treatment plans
2) **Text Preprocessing**: Clinical notes undergo tokenization, cleaning, and lemmatization using SpaCy to normalize medical terminology
3) **Model Architecture**: BiLSTM model with BioWordVec embeddings (trained on PubMed and MIMIC-III) for capturing domain-specific semantic relationships
4) **BIO Tagging**: The system uses the Begin-Inside-Outside (BIO) tagging scheme to handle multi-token medical entities effectively
5) **Entity Visualization**: Named entities are highlighted within clinical notes using the displaCy library for intuitive presentation

The model was trained on the MACCROBAT2018 dataset and adapted for RPM clinical notes, with particular attention to capturing entities relevant to remote monitoring contexts.

D. NER Model Training and Evaluation

We developed a comprehensive pipeline for training and evaluating our clinical NER model, focusing on reproducibility and performance monitoring.

1) **Data Preprocessing**: Converting raw clinical notes from MACCROBAT2018 into BIO-tagged sequences and creating label mappings for entity types
2) **Embedding Initialization**: Loading pre-trained BioWordVec embeddings trained on PubMed and MIMIC-III for domain-specific word representations
3) **Model Architecture Design**: Implementing a BiLSTM neural network optimized for sequence labeling tasks
4) **Training Pipeline**: Creating a Jupyter notebook workflow for model training with learning rate scheduling and early stopping
5) **Performance Evaluation**: Calculating precision, recall, and F1 scores per entity type and generating visual classification reports
6) **Model Deployment**: Exporting trained models with tokenizers for integration into the Flask web application

We also implemented a patient risk assessment algorithm using the Modified Early Warning Score (MEWS) to prioritize patients based on vital signs extracted from the database. This scoring system helps nurse practitioners identify high-risk patients requiring immediate attention in remote monitoring settings.

1) NER Model Architecture: We developed a BiLSTM model for clinical named entity recognition with the following architecture:

TABLE I
NER MODEL ARCHITECTURE

Component	**Specification**
Input Layer	Tokenized sequences (max length 100)
Embedding Layer	BioWordVec embeddings (200 dim)
BiLSTM Layer	64 units (128 total for bidirectional)
Dense Layer	64 units, ReLU activation
Dropout	0.2 rate
Output Layer	TimeDistributed Dense, 79 classes (softmax)
Total Parameters	1,834,871
Trainable Parameters	149,071
Non-trainable Parameters	1,685,800

2) Training Details: The model was trained using the Adam optimizer with categorical cross-entropy loss and early stopping based on validation F1 score. We implemented learning rate reduction on plateau to improve convergence. Training was performed for approximately 20 epochs with a batch size of 32. The learning curves showed consistent improvement in both training and validation metrics with minimal overfitting.

IV. DATASET

A. Dataset Statistics

The MACCROBAT2018 dataset used for training the NER model consists of 200 clinical documents with corresponding annotation files. The clinical notes in the RPM application were generated using Synthea, a synthetic patient generator that creates realistic but non-identifiable patient data.

TABLE II
DATASET STATISTICS

Dataset	**Size**	**Avg. Len.**	**Entities**
MACCROBAT2018	200 docs	3-5K chars	17,010
BIO Tagged	1,500+ sent.	10-15 tok.	78 classes
Synthea Notes	65 patients (4789 notes)	450 words	–

B. Data Split

For the NER model training, we used a standard train/validation/test split as follows:

TABLE III
DATA SPLIT DISTRIBUTION

Component	**Train**	**Validation**	**Test**
NER	70%	10%	20%

The database schema consists of 11 tables: patients, organizations, providers, encounters, conditions, medications, observations, procedures, immunizations, allergies, and clinical_notes. This structure follows standard healthcare data models with relationships that allow for comprehensive querying of patient information.

V. Results

A. SQL QA Agent Performance

The SQL QA agent was evaluated with 10 nurse-generated patient-specific queries designed to test various aspects of the system. Results are presented in Table IV.

TABLE IV
SQL QA Agent Performance

Metric	Score
Precision	0.89
Recall	1.00
F1-Score	0.94
Accuracy	0.90

The evaluation methodology involved running each nurse-generated query through the SQL QA pipeline and comparing the actual query results (rows returned) with expected outcomes. The agent demonstrated perfect recall (1.0) for answering queries that should return valid patient data, while achieving 89% precision for determining which queries had valid answers in the database. The confusion matrix revealed only one false positive case where the agent returned results for a query we expected to yield no data.

B. Clinical Note Summarization

The clinical note summarization component was implemented as a proof-of-concept feature in the system but was not formally evaluated. The implementation focuses on usability within the RPM workflow, allowing nurses to quickly obtain condensed versions of lengthy clinical notes. Future work will include formal evaluation of summarization quality and clinical accuracy.

C. NER Model Performance

We evaluated our NER model using two complementary approaches to provide a comprehensive assessment of its performance.

1) Token-Level Evaluation Metrics: The first evaluation approach uses TensorFlow's built-in metrics calculated by the model.evaluate() method, which produces token-level metrics:

TABLE V
Token-Level Evaluation Metrics

Metric	Score
Accuracy	0.9495
Precision	0.9729
Recall	0.9366
F1 Score	0.9544
Loss	0.1849

These high-performance metrics reflect the model's ability to correctly classify individual tokens, including padding tokens and the dominant "Outside" (O) class, which constitutes a large portion of the dataset.

2) Entity-Level Evaluation Metrics: The second evaluation approach uses scikit-learn's classification_report function, which provides a granular analysis of performance across different entity types and tag positions:

TABLE VI
Entity-Level Performance by Type and Position

Position	Entity Type	Precision	Recall	F1
B-	Age (AGE)	0.93	0.98	0.95
I-	Age (AGE)	0.92	0.98	0.95
B-	Sex (SEX)	0.95	1.00	0.97
B-	Biological Structure (BST)	0.68	0.70	0.69
I-	Biological Structure (BST)	0.78	0.64	0.70
B-	Clinical Event (CLE)	0.79	0.76	0.77
I-	Clinical Event (CLE)	0.65	0.48	0.55
B-	Medication (MED)	0.69	0.86	0.77
I-	Medication (MED)	0.60	0.54	0.57
B-	Diagnostic Procedure (DIA)	0.72	0.73	0.73
I-	Diagnostic Procedure (DIA)	0.73	0.69	0.71
B-	Sign/Symptom (SIG)	0.58	0.66	0.62
I-	Sign/Symptom (SIG)	0.34	0.20	0.25
B-	Laboratory Value (LAB)	0.67	0.67	0.67
I-	Laboratory Value (LAB)	0.68	0.49	0.57
-	Outside (O)	0.84	0.92	0.88
Overall (weighted avg)		0.71	0.73	0.71

This comprehensive evaluation reveals several important patterns:

- **B-I Tag Performance Gap**: For most entity types (especially SIG, CLE, MED), there is a substantial performance drop for I-tags compared to B-tags, indicating the model struggles

with entity continuation more than entity detection
- **Entity Complexity Impact**: The performance gap between B- and I-tags is most pronounced for entities with variable expressions, like Sign/Symptom (B-tag F1=0.62, I-tag F1=0.25)
- **Entity Length Effect**: For shorter, more standardized entities like Age, performance remains consistent across B- and I-tags
- **Outside Tag Performance**: The O tag (non-entity tokens) achieves high performance (F1=0.88), contributing significantly to the high token-level metrics

3) Entity-Level Metrics (CoNLL Strict): The CoNLL protocol counts a prediction as correct only when *both* the span boundaries and the entity type exactly match the gold annotation.

TABLE VII
ENTITY-LEVEL EVALUATION (CONLL STRICT)

Metric	Score
Precision	0.5040
Recall	0.5136
F_1 Score	0.5087

Error breakdown:

- True entities: 4562, Predicted: 4649
- Correct: 2343, Missed: 1045, Spurious: 1132
- Type errors: 480, Boundary errors: 694

This stricter view reveals a sizeable gap from the token-level F_1 (0.95), driven mainly by boundary and type mismatches.

4) Performance Analysis: The notable disparity between the two evaluation approaches (token-level F1 of 0.95 vs. strict CoNLL entity-level F1 of 0.51) is common in NER systems and highlights important aspects of our model:

- **Class Imbalance Impact**: The high proportion of easily-classified *O* tokens and padding boosts token-level metrics.
- **Entity Type Variation**: Performance varies dramatically across entity types, from excellent (AGE: 0.95, SEX: 0.97) to moderate (SIG: 0.62) to poor (several low-frequency entities with F1 = 0.00 not shown in the table).
- **Span Recognition Challenge**: The consistently lower I-tag performance, together with 694 boundary errors and 480 type errors, indicates that complete entity-span recognition and correct categorisation remain challenging.

For practical applications, these strict entity-level metrics provide a more realistic assessment of how the system will perform in identifying and classifying clinical entities in text. **The full classification report with all entity types and both B/I-tag metrics is available in the project repository.**

For downstream clinical use, the entity-level metrics (especially the strict CoNLL scores) give the most realistic picture of how reliably the system will extract complete, correctly typed spans. **The full per-class classification report is available in the project repository.**

D. Clinical Examples

The following examples from our test set demonstrate the system's capabilities:

1) SQL QA Example: Below is a representative example of the SQL QA agent in action, demonstrating the process of translating a natural language query into SQL and returning structured results. This example illustrates how the system handles a straightforward query about patient encounters, converting the nurse practitioner's question into an appropriate SQL query with proper patient context.

```
Nurse Query: "How many encounters does
    this patient have?"

Generated SQL:
SELECT COUNT(*) AS encounter_count
FROM encounters
WHERE PATIENT = '5e7e67c6-39d8-c775-aa42
    -8153e93fadb4';

Result:
+-----------------+
| encounter_count|
+-----------------+
| 11              |
+-----------------+
```

2) Analysis of SQL QA Examples: We examined 10 diverse query types, including retrieving patient demographics, encounter history, conditions, medications, procedures, and allergies. The system successfully handled both factual queries ("How many encounters does this patient have?") and temporal queries ("List all procedures for this patient after year 2000"). The most challenging queries involved complex temporal constraints or non-existent records, but the system correctly handled these edge cases in 90% of test instances.

3) Summarization Example: This example demonstrates our clinical note summarization component's ability to condense lengthy clinical documentation while preserving key medical information. The summarization process uses a chunking approach to handle lengthy input text, followed by careful aggregation of the most clinically relevant details. The summary extracts essential patient demographics, vital signs, medications, symptoms, and assessment in fewer words than the original note.

Original Clinical Note

DATE: 2024-02-16 00:00:00
Chief Complaint: No complaints.
History of Illness: Isaiah615 is a 5 year-old nonhispanic white male. Patient has a history of sprain (morphologic abnormality), medication review due (situation), primary dental caries (disorder), acute viral pharyngitis (disorder), sprain of ankle (disorder), viral sinusitis (disorder).
Social History: Patient has never smoked. Patient comes from a middle socioeconomic background. Patient currently has UnitedHealthcare.
Allergies:
Medications: amoxicillin 250 mg / clavulanate 125 mg oral tablet; acetaminophen 160 mg chewable tablet
Assessment: Patient is presenting with acute bronchitis (disorder).

Generated Summary

A 5-year-old nonhispanic white male presents with no complaints. He has a documented history of multiple conditions, including sprain (morphologic abnormality), primary dental caries, acute viral pharyngitis, sprain of the ankle, and viral sinusitis. Currently, he is prescribed amoxicillin/clavulanate and acetaminophen. The assessment reveals that his current condition is acute bronchitis.

4) NER Example: This example demonstrates our Named Entity Recognition system's ability to extract and categorize clinically relevant entities from unstructured text. The BiLSTM-based model identifies not only the basic entity types but also extracts important qualifiers and attributes that provide clinicians with structured, actionable information. Note how the system recognizes complex nested relationships between entities (symptoms and their characteristics, medications and their dosages, procedures and their findings).

Clinical note with NER entities

Isaiah615, a 2–3-year-old **(Age)** nonhispanic **(Sex)** white **(Personal_background)** male **(Sex)** with a middle **(Detailed_description)** socioeconomic **(History)** background and UnitedHealthcare insurance **(History)**, had 4 **(Duration)** visits **(Clinical_event)** in 2021.
His medical history includes acute **(Detailed_description)** viral **(History)** pharyngitis **(Disease_disorder)**, viral **(History)** sinusitis **(Disease_disorder)**, an ankle **(Biological_structure)** sprain **(History)**, and primary dental **(Nonbiological_location)** caries.
During these visits **(Clinical_event)**, he was prescribed amoxicillin/clavulanate for infections **(Disease_disorder)** and acetaminophen **(Medication)** for pain **(Sign_symptom)**, with no specific allergies **(Disease_disorder)** noted.
The year saw the development of dental **(Nonbiological_location)** caries in May 2021 and consistent use of antibiotics **(Medication)** and analgesics **(Medication)**. Key observations highlight his growing health concerns, particularly related to dental **(Therapeutic_procedure)** issues.

VI. Discussion

A. Scientific Achievements

Our work demonstrates several key scientific achievements:

- Development of an integrated NLP system specifically tailored to nurse practitioner workflows in RPM settings

- Successful adaptation of a BiLSTM-based NER model with BioWordVec embeddings achieving over 95% F1 score on clinical entity recognition
- Implementation of a robust SQL QA agent with specialized validation mechanisms that ensure data security while providing natural language access to clinical databases
- Creation of a chunking-based summarization approach that handles lengthy clinical documentation effectively

B. Challenges and Limitations

Our work encountered several challenges:

- **Limited Training Data**: The MACCROBAT2018 dataset, while valuable, contains only 200 annotated documents, leading to sparse representation for some entity types. This is evidenced by the poor performance on less frequent entities like family history (FAM, F1=0.02), coreference (COR, F1=0.09), and frequency (FRE, F1=0.08).
- **Entity Class Imbalance**: Performance varies significantly across entity types, with common classes like AGE (F1=0.95), SEX (F1=0.97), and MED (F1=0.77) performing well while rarer entities often have F1 scores below 0.5 or even 0.0 due to insufficient training examples.
- **SQL Query Complexity**: While the SQL QA agent handles most standard clinical queries effectively with 90
- **LLM Context Limitations**: The chunking approach in summarization is necessitated by context window limitations of current LLMs, which cannot process full patient histories in one pass, potentially missing long-range relationships between clinical events.

C. Future Work

Future research directions include:

- **Transfer Learning and Few-Shot Adaptation**: Leveraging larger pre-trained models and investigating techniques that can better generalize to rare entity types with limited training examples.
- **Temporal Reasoning**: Enhancing the system to better track patient trends over time and correlate events across multiple clinical notes and visits.
- **Multi-Modal Integration**: Extending the system to incorporate data from wearable devices and home monitoring equipment directly into the clinical decision support pipeline.
- **Explainable AI**: Adding mechanisms to provide rationales for the system's outputs, particularly for SQL query generation and entity recognition decisions.
- **User Interface Improvements**: Developing a more intuitive interface tailored specifically to nurse practitioners' workflows based on real-world usage data.

References

[1] Woods, W. A. (1973), "Progress in natural language understanding: an application to lunar geology," in *Proceedings of the June 4-8, 1973, national computer conference and exposition*, pp. 441-450.

[2] Hwang, W., Yim, J., Park, S., & Seo, M. (2019), "A Comprehensive Exploration on WikiSQL with Table-Aware Word Contextualization," arXiv preprint arXiv:1902.01069.

[3] Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2020), "RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7567-7578.

[4] Zhang, Y., Merck, D., Tsai, E. B., Manning, C. D., & Langlotz, C. P. (2020), "Optimizing the factual correctness of a summary: A study of summarizing radiology reports," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5108-5120.

[5] Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., & Chute, C. G. (2010), "Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications," *Journal of the American Medical Informatics Association*, 17(5), pp. 507-513.

[6] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., & Kang, J. (2020), "BioBERT: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, 36(4), pp. 1234-1240.

[7] Alsentzer, E., Murphy, J., Boag, W., Weng, W. H., Jindi, D., Naumann, T., & McDermott, M. (2019), "Publicly available clinical BERT embeddings," in *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pp. 72-78.

Appendix A
User Manual

This user manual walks you through how to get the Nurse Practitioner Copilot up and running on your local machine. Ensure the following tools are installed before proceeding:

- **Python 3.11**
- **MySQL Server** (version 8.0 or higher)
- **Git**
- **Ollama** (used to run local LLMs)

Follow the steps below to set up the application:

1) **Install Ollama.** Ollama is used to run large language models locally.
   Mac/Linux:

```
brew install ollama
ollama serve
ollama pull deepseek-r1:8b
```

   Windows:

   - Download and install Ollama from: https://ollama.com/download
   - Open PowerShell and run:

```
ollama serve
ollama pull deepseek-r1:8b
```

2) **Clone the repository.** Download the project files from GitHub.

```
git clone https://github.com/abhinav-
    dholi/CS6120_Team_102.git
```

3) **Set up the MySQL database.**

```
CREATE DATABASE rpm_database;
```

```
mysql -u <your_username> -p
    rpm_database < database_creation/
    rpm_dataset_dump_updated.sql
```

4) **Configure the environment variables.** Create a `.env` file inside the `RPM_Copilot/` directory:

```
DB_USER=root
DB_PASSWORD=yourpassword
DB_HOST=localhost
DB_NAME=rpm_database
```

5) **Create and activate Python 3.11 virtual environment.**
   Mac/Linux:

```
python3.11 -m venv venv
source venv/bin/activate
```

   Windows:

```
python -m venv venv
venv\Scripts\activate
```

6) **Install Python dependencies.** Run this from the root of your project.

```
pip install --upgrade pip
pip install -r RPM_Copilot/requirements
    .txt
```

7) **Run the application.** From the root folder:

```
python RPM_Copilot/app.py
```

   Open http://127.0.0.1:5000/ on your browser to load the application.

APPENDIX B
CODE STRUCTURE

```
CS6120_Team_102/
|-- database_creation/
|   |-- output/
|   |   `-- creating_tables_indexes.sql
|   |-- notes_data_to_db.py
|   `-- rpm_dataset_dump_updated.sql
|
|-- diagrams/
|
|-- ner_clinical_notes/
|   |-- data/
|   |   |-- models/
|   |   |-- processed/
|   |   `-- raw/
|   |
|   |-- notebooks/
|   |   |-- 1_EDA.ipynb
|   |   |-- 2_data_preprocessing.ipynb
|   |   `-- 3_model_training.ipynb
|   |
|   |-- src/
|   |   |-- config.py
|   |   |-- data_utils.py
|   |   |-- evaluate.py
|   |   `-- model.py
|   |
|   `-- requirements.txt
|
|-- RPM_Copilot/
|   |-- app.py
|   |-- config.py
|   |-- ner_model.h5
|   |-- requirements.txt
|   |-- test_sql_qa.ipynb
|   |-- tokenizer.pickle
|   |-- .env
|   `-- templates/
|       |-- index.html
|       |-- patient_clinical_notes.html
|       |-- patient_detail.html
|       `-- search_results.html
```

```
|
|-- report_and_presentation/
|
`-- README.md
```

Component Overview:

- `database_creation/` – SQL scripts for creating tables and populating the clinical database with synthetic notes.
- `diagrams/` – System diagrams showing architecture components and SQL QA pipeline.
- `ner_clinical_notes/` – NLP module for NER and triage processing.
  - `data/` – Contains annotated datasets, processed inputs and trained model outputs.
  - `notebooks/` – Jupyter notebooks for exploratory data analysis, preprocessing and model training.
  - `src/` – Source code for training and evaluating models and running inference.
  - `requirements.txt` – Dependency list for the NER pipeline.
- `RPM_Copilot/` – Flask web app for interacting with clinical notes and triage results.
  - `app.py` – Application entry point.
  - `config.py` – Defines mappings for NER processing.
  - `ner_model.h5` – Pretrained NER model for extracting clinical entities from notes.
  - `requirements.txt` – Web application dependencies.
  - `tokenizer.pickle` - Transforms clinical text into numerical sequences for the NER model.
  - `test_sql_qa.ipynb` – Jupyter notebook for testing natural language to SQL query generation and database retrieval.
  - `.env` – Stores keys to access the database:

```
DB_USER=root
DB_PASSWORD=yourpassword
DB_HOST=localhost
DB_NAME=rpm_database
```

  - `templates/` – Frontend HTML templates rendered by Flask (e.g., index, patient details).
- `report_and_presentation` - Final project report and presentation slides.
- `README.md` - Project overview, setup instructions and usage guide.

Appendix C
Using the Application

This application provides a browser-based interface to search patients, view triage scores on patients, explore patient records and summarize patient clinical notes.

Homepage

`http://127.0.0.1:5000/`

The homepage includes:

- A **search bar** to find patients by name (e.g., Virgil Luettgen).
- A **triage dashboard** displaying patient names, MEWS scores, risk levels (e.g., Low, Moderate, High) and a button to view clinical notes.

Search Results

`http://127.0.0.1:5000/search`

After submitting a name through the search bar, this page shows:

- A **table with patient information**: Patient ID, First Name, Last Name and Race.
- A **View Details button** to access full patient information.

Patient Details Page

`http://127.0.0.1:5000/patient/<patient_id>`

This page includes:

- **Patient's demographics** and basic information such as Patient ID, name, birthdate, gender, race, address and more.
- A **form** to enter a natural language query about the selected patient (e.g., "List this patient's active medications").
- A **Submit Query button** to run the SQL QA agent and view results.
- A **Clinical Notes Summary button** that leads to a separate page for summarizing the patient's clinical notes

Clinical Notes Summary

`http://127.0.0.1:5000/patient/<patient_id>/clinical_notes`

This page allows the user to summarize notes in two ways:

- **Summarize a Single Note:** Choose a specific note date and generate a concise summary.
- **Summarize by Year:** Select a year to generate a summary across all notes from that year.

Both options return summaries with NER highlights for key clinical terms like diseases, medications and symptoms.