UNIVERSITÉ DE LORRAINE

MASTER'S THESIS

Transforming Unstructured Biomedical Text into Actionable Knowledge Graphs: A Multi-Model Approach for Healthcare Information Extraction

Author:
Oyetunji Daniel ABIOYE

Supervisor:
Maziyar PANAHI

A thesis submitted in fulfillment of the requirements for the degree of Master of Science

in the

IDMC
L'Institut des Sciences du Digital Management & Cognition

August 18, 2025

Declaration of Authorship

I, Oyetunji Daniel ABIOYE, declare that this thesis titled, "Transforming Unstructured Biomedical Text into Actionable Knowledge Graphs: A Multi-Model Approach for Healthcare Information Extraction" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at Université de lorraine.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UNIVERSITÉ DE LORRAINE

Abstract

IDMC
L'Institut des Sciences du Digital Management & Cognition

Master of Science

Transforming Unstructured Biomedical Text into Actionable Knowledge Graphs: A Multi-Model Approach for Healthcare Information Extraction

by Oyetunji Daniel ABIOYE

The exponential growth of biomedical text data presents significant challenges and opportunities for extracting actionable knowledge that can advance healthcare and biomedical research. Unstructured biomedical documents, including research abstracts and clinical notes, contain vast amounts of valuable information not readily accessible through traditional structured databases. This thesis proposes a comprehensive, multi-model pipeline designed to transform unstructured biomedical text into structured, actionable knowledge graphs.

Leveraging advanced natural language processing (NLP) techniques, the pipeline integrates specialized models for named entity recognition (GLiNER), entity linking to standardized medical knowledge bases (UMLS via SciSpacy), and relationship extraction using large language models (LLMs). Specifically, domain-specific models like MedGemma are compared against general-purpose models such as Gemma to evaluate their effectiveness in accurately extracting biomedical relationships. The pipeline's modular design ensures efficient processing, robust entity linking, and precise relationship extraction, addressing critical challenges such as synonymy, ambiguity, and the complexity of biomedical language.

The extracted data populates a knowledge graph constructed using Neo4j, allowing efficient storage, querying, and retrieval of complex biomedical relationships. The graph schema and query generation processes are carefully designed to ensure data integrity and scalability. Comprehensive evaluations using the BioRED dataset demonstrate the pipeline's high accuracy in entity recognition and relationship extraction, underscoring its utility in real-world clinical and research applications.

Overall, this thesis contributes to biomedical NLP by delivering an integrated solution that systematically converts unstructured biomedical literature into actionable, structured knowledge, facilitating accelerated research discovery and enhanced clinical decision-making.

Acknowledgements

First and foremost, I would like to express my heartfelt gratitude to my supervisor, Maziyar Panahi, whose invaluable guidance, continuous support, and expert advice significantly enriched my research journey. Their insightful feedback and encouragement have been crucial in shaping this thesis and my academic growth.

I extend my sincere appreciation to the faculty and staff at L'Institut des Sciences du Digital Management & Cognition (IDMC) at Université de Lorraine, for providing an inspiring academic environment and the necessary resources that facilitated this research.

My deepest thanks to the community and contributors behind the biomedical datasets and open-source tools that formed the backbone of this project. Their dedication and open collaboration have profoundly advanced research possibilities in biomedical natural language processing and knowledge graph development.

Special thanks go to my peers and colleagues who offered their time, discussions, and valuable perspectives throughout the development of this work. Your camaraderie and intellectual engagement have greatly enriched my experience.

Lastly, I am eternally grateful to my family and friends for their unwavering support, encouragement, and patience throughout my academic journey. Their belief in me has been my greatest motivation.

Thank you all for being a vital part of this journey.

Contents

Declaration of Authorship iii

Abstract v

Acknowledgements vii

1 Introduction 1
- 1.1 Background and Context . . . 1
  - 1.1.1 Biomedical Text and Knowledge Sources . . . 1
  - 1.1.2 Challenges with Unstructured Biomedical Text . . . 1
- 1.2 Research Motivation . . . 2
  - 1.2.1 The Need for Structured Information Extraction . . . 2
  - 1.2.2 Graph Databases for Biomedical Knowledge . . . 3
- 1.3 Research Question and Objectives . . . 4
  - 1.3.1 Primary Research Question . . . 4
  - 1.3.2 Specific Objectives . . . 4
- 1.4 Ethical Considerations and Data Privacy . . . 5
  - 1.4.1 Responsible Use of Biomedical Data . . . 5
- 1.5 Thesis Contributions . . . 5

2 Literature Review and Theoretical Background 7
- 2.1 Natural Language Processing in Biomedicine . . . 7
  - 2.1.1 Evolution of Biomedical NLP . . . 7
  - 2.1.2 Current State-of-the-Art Approaches . . . 7
- 2.2 Named Entity Recognition in Biomedical Text . . . 8
  - 2.2.1 Traditional NER Methods . . . 8
  - 2.2.2 Deep Learning Approaches for Biomedical NER . . . 9
  - 2.2.3 Biomedical Language Models . . . 9
- 2.3 Entity Linking Techniques . . . 10
  - 2.3.1 Knowledge Base Linking Methods . . . 10
  - 2.3.2 UMLS and MeSH Knowledge Bases . . . 11
  - 2.3.3 SciSpacy's TF-IDF Character N-gram Matching . . . 12
  - 2.3.4 Challenges in Medical Entity Linking . . . 12
- 2.4 Relationship Extraction and Knowledge Graphs . . . 14
  - 2.4.1 Relationship Extraction Methods . . . 14
    - Rule-Based Approaches . . . 14
    - Machine Learning Techniques . . . 14
    - Large Language Models for Relationships . . . 15
  - 2.4.2 Knowledge Graph Construction . . . 16
    - Graph Database Technologies . . . 16
    - Neo4j and Cypher Query Language . . . 17
  - 2.4.3 Biomedical Knowledge Graph Applications . . . 18
- 2.5 Related Work Summary and Research Gap . . . 19

3 Methodology **21**
3.1 System Architecture and Pipeline . . . 21
3.1.1 Overall System Design . . . 21
3.1.2 Component Integration Flow . . . 22
3.2 Data and Preprocessing . . . 23
3.2.1 Biomedical Literature Dataset and Preparation . . . 23
3.3 Entity Recognition and Linking . . . 24
3.3.1 GLiNER Model for Medical Entity Recognition . . . 24
3.3.2 UMLS Entity Linking with SciSpacy . . . 25
3.3.3 Confidence Scoring and Abbreviation Detection . . . 25
3.4 Relationship Extraction . . . 26
3.4.1 Model Selection and Prompt Engineering . . . 26
3.4.2 Output Parsing and Validation . . . 27
3.5 Knowledge Graph Construction . . . 28
3.5.1 Graph Schema Design . . . 28
3.5.2 Cypher Query Generation and Storage . . . 30
3.6 Implementation Optimization . . . 32
3.6.1 Parallel Processing and Resource Management . . . 32

4 Results, Evaluation and Discussion **35**
4.1 Experimental Setup . . . 35
4.1.1 BioRED Dataset and Evaluation Metrics . . . 35
4.1.2 Model Configurations and Prompt Strategies . . . 36
4.2 Named Entity Recognition Performance . . . 37
4.2.1 GLiNER Results: Threshold Impact and Matching Strategies . . . 37
4.2.2 Error Analysis and Entity Boundary Challenges . . . 38
4.3 Relationship Extraction Performance . . . 39
4.3.1 Gemma vs MedGemma: Strategy Comparison and Low F1 Analysis . . . 39
4.3.2 Model Limitations and Output Quality Issues . . . 40
4.4 End-to-End Pipeline Evaluation . . . 42
4.4.1 Processing Efficiency and Scalability . . . 42
4.5 Discussion . . . 43
4.5.1 Key Findings and Clinical Applicability . . . 43

5 Conclusions and Future Work **45**
5.1 Summary of Contributions . . . 45
5.1.1 Technical Contributions . . . 45
5.1.2 Theoretical Contributions . . . 45
5.2 Future Research Directions . . . 45
5.2.1 Model Scale Improvements . . . 46
5.2.2 Expanding Beyond UMLS to Other Medical Knowledge Bases . . . 46
5.3 Final Remarks . . . 46

A Comprehensive Evaluation Results and Statistical Analysis **47**
A.1 BioRED Dataset Evaluation Overview . . . 47
A.2 Named Entity Recognition (NER) Performance . . . 47
A.2.1 GLiNER Threshold Sensitivity Analysis . . . 47
A.2.2 Matching Strategy Performance Comparison . . . 48
A.2.3 Entity Type-Specific Performance . . . 48
A.3 Relationship Extraction Performance . . . 48

A.3.1 Model Configuration Comparison . . . 48
A.3.2 Detailed Error Analysis . . . 48
Output Format Issues . . . 49
Model-Specific Observations . . . 49
A.4 Prompting Strategy Analysis . . . 49
A.4.1 Basic Prompting . . . 49
A.4.2 Few-Shot Prompting . . . 49
A.4.3 Structured JSON Prompting . . . 49
A.5 Statistical Significance Tests . . . 49
A.6 Performance by Relation Type . . . 49
A.7 Computational Performance Metrics . . . 50
A.7.1 Processing Time Analysis . . . 50
A.8 Conclusions from Evaluation . . . 50
A.9 Recommendations for Future Work . . . 50

B Technical Implementation and Reproducibility Guide 51
B.1 System Requirements and Dependencies . . . 51
B.1.1 Hardware Requirements . . . 51
B.1.2 Software Dependencies . . . 51
B.2 GLiNER Biomedical Model Configuration . . . 51
B.2.1 Model Initialization . . . 51
B.2.2 Threshold Configuration . . . 52
B.2.3 Optimization Parameters . . . 52
B.3 Relationship Extraction Prompt Templates . . . 52
B.3.1 Basic Prompting Strategy . . . 52
B.3.2 Few-Shot Prompting Strategy . . . 53
B.3.3 Structured JSON Prompting Strategy . . . 54
B.4 Neo4j Graph Schema and Cypher Templates . . . 54
B.4.1 Graph Schema Design . . . 54
B.4.2 Entity Creation Cypher Template . . . 55
B.4.3 Relationship Creation Cypher Template . . . 55
B.4.4 Query Templates for Knowledge Retrieval . . . 55
B.5 SciSpacy Entity Linking Configuration . . . 56
B.5.1 Pipeline Setup . . . 56
B.5.2 Entity Linking Process . . . 56
B.6 Model Training and Deployment . . . 57
B.6.1 Environment Setup . . . 57
B.6.2 Running the Pipeline For GLiNER Evaluation . . . 57
B.6.3 Running the Pipeline For Gemma Evaluation . . . 57
B.7 Performance Optimization . . . 57
B.7.1 Caching Strategy . . . 57
B.7.2 Parallel Processing . . . 57
B.8 Reproducibility Checklist . . . 58
B.9 Troubleshooting Common Issues . . . 58
B.9.1 Memory Issues . . . 58
B.9.2 UMLS Licensing . . . 58
B.9.3 Neo4j Connection . . . 58

C Error Analysis and Failure Case Studies **59**
C.1 Overview of System Failures 59
C.2 Relationship Extraction Failure Analysis 59
C.2.1 Quantitative Failure Rates 59
C.2.2 Common Failure Patterns 60
Entity Hallucination 60
Incorrect Entity Matching 60
Relation Type Confusion 60
C.3 Specific Failure Case Studies 60
C.3.1 Case Study 1: Complex Medical Relationships 60
C.3.2 Case Study 2: Drug Interaction Scenario 61
C.3.3 Case Study 3: Biochemical Pathway 61
C.4 Prompt Strategy Effectiveness Analysis 62
C.4.1 Basic Prompting Failures 62
C.4.2 Few-Shot Prompting Failures 62
C.4.3 Structured JSON Prompting Failures 62
C.5 Domain-Specific Model Underperformance 63
C.5.1 MedGemma vs Gemma Comparison 63
C.5.2 Hypotheses for Domain Model Underperformance 63
C.6 Error Cascading Effects 63
C.6.1 NER to Relationship Extraction 63
C.6.2 Example of Error Propagation 63
C.7 Computational Error Analysis 64
C.7.1 Memory and Resource Failures 64
C.7.2 Performance Degradation Patterns 64
C.8 Recommendations Based on Error Analysis 64
C.8.1 Immediate Improvements 64
C.8.2 Architectural Changes 64
C.8.3 Training and Fine-tuning 64
C.9 Conclusion 65

Bibliography **67**

List of Figures

3.1 System architecture pipeline showing the transformation of biomedical text into a knowledge graph 21

List of Tables

4.1 NER Evaluation Summary - GLiNER-BioMed Performance Across Configurations . . . 38
4.2 Relationship Extraction Evaluation Summary - Gemma vs MedGemma Performance . . . 40

A.1 GLiNER Performance at Different Confidence Thresholds (Exact Matching) . . . 47
A.2 GLiNER Performance Across Different Matching Strategies (Threshold=0.5) . . . 48
A.3 Relationship Extraction Performance Across Model Configurations . . . 48
A.4 Average Processing Time per Document . . . 50

C.1 Relationship Extraction Failure Rates by Model Configuration . . . 59
C.2 Model Output Comparison for Biochemical Pathway Extraction . . . 62
C.3 Comparative Error Analysis: MedGemma vs Gemma . . . 63
C.4 Error Rates by Document Complexity . . . 64

List of Abbreviations

AI	**A**rtificial **I**ntelligence
API	**A**pplication **P**rogramming **I**nterface
BERT	**B**idirectional **E**ncoder **R**epresentations from **T**ransformers
BiLSTM	**Bi**directional **L**ong **S**hort-**T**erm **M**emory
BioBERT	**Bio**medical **BERT**
BioRED	**Bio**medical **R**elation **E**xtraction **D**ataset
CNN	**C**onvolutional **N**eural **N**etwork
CPU	**C**entral **P**rocessing **U**nit
CRF	**C**onditional **R**andom **F**ields
CUI	**C**oncept **U**nique **I**dentifier
DNA	**D**eoxyribonucleic **A**cid
EHR	**E**lectronic **H**ealth **R**ecords
FN	**F**alse **N**egatives
FP	**F**alse **P**ositives
GLiNER	**G**eneralist and **L**ightweight **N**amed **E**ntity **R**ecognition
GPT	**G**enerative **P**re-trained **T**ransformer
GPU	**G**raphics **P**rocessing **U**nit
HIPAA	**H**ealth **I**nsurance **P**ortability and **A**ccountability **A**ct
ICD	**I**nternational **C**lassification of **D**iseases
IDMC	**I**nstitut des **S**ciences du **D**igital **M**anagement & **C**ognition
IE	**I**nformation **E**xtraction
JSON	**J**avaScript **O**bject **N**otation
KB	**K**nowledge **B**ase
LLM	**L**arge **L**anguage **M**odel
LSTM	**L**ong **S**hort-**T**erm **M**emory
MedGemma	**Med**ical **Gemma**
MeSH	**Me**dical **S**ubject **H**eadings
ML	**M**achine **L**earning
NER	**N**amed **E**ntity **R**ecognition
NLP	**N**atural **L**anguage **P**rocessing
PMID	**P**ubMed **ID**entifier
PubMed	**Pub**lic **Med**line
PubMedBERT	**PubMed BERT**
RAM	**R**andom **A**ccess **M**emory
RNN	**R**ecurrent **N**eural **N**etwork
RxNorm	**Rx**Norm (Drug Naming System)
SciBERT	**Sci**entific **BERT**
SNOMED CT	**S**ystematized **No**menclature of **Med**icine **C**linical **T**erms
SOTA	**S**tate-**o**f-**t**he-**A**rt
SVM	**S**upport **V**ector **M**achine
TF-IDF	**T**erm **F**requency-**I**nverse **D**ocument **F**requency
TP	**T**rue **P**ositives

UMLS	**U**nified **M**edical **L**anguage **S**ystem
VRAM	**V**ideo **RAM**

For/Dedicated to/To my…

Chapter 1

Introduction

1.1 Background and Context

1.1.1 Biomedical Text and Knowledge Sources

The biomedical domain generates vast amounts of textual data through multiple channels including electronic health records (EHRs), scientific literature, and research databases (Kong, 2019). This biomedical text comes in both **structured** and **unstructured** formats. Structured data refers to information stored in predefined fields (e.g. patient demographics, clinical codes, genomic sequences, chemical formulas) that can be readily queried. In contrast, **unstructured data** consists of free-form text such as clinical narratives, research abstracts, full-text articles, case reports, and experimental descriptions. Free-text biomedical documents are a natural and comprehensive way for healthcare providers and researchers to communicate complex medical knowledge, often capturing nuanced relationships between biomedical entities that do not fit neatly into structured fields (Klug et al., 2024). For example, a research abstract may describe novel drug-disease interactions or a physician's note may document symptom severity in richer detail than any standardized code.

Notably, unstructured content constitutes the majority of biomedical documentation. By some estimates, **up to 80% of healthcare and biomedical data is unstructured** text (Kong, 2019). This includes clinical notes, research publications (PubMed contains over 35 million citations), experimental protocols, and case studies that do not reside in tabular databases but in narrative form. Such unstructured biomedical text is immensely valuable, as it provides comprehensive documentation of clinical reasoning, experimental findings, and scientific discoveries. However, because it lacks a predefined format, unstructured data is not directly usable by computers or standard query tools. Healthcare organizations and researchers are therefore faced with the challenge of leveraging these abundant textual documents to improve patient care, accelerate research, and enable evidence-based decision-making.

1.1.2 Challenges with Unstructured Biomedical Text

Working with unstructured biomedical text is inherently challenging. Free-text biomedical documents are **difficult for automated systems to interpret** due to their lack of consistent structure and the complexities of scientific and clinical language (Yazdani, Stepanov, and Teodoro, 2025). Unlike structured database entries, biomedical text may contain context-dependent information, ambiguous terminology, and diverse writing styles across different domains (clinical, research, pharmaceutical). Indeed, biomedical text often includes **domain-specific abbreviations, technical jargon, complex chemical names, and variable nomenclature**, all of which increase the difficulty of parsing and analyzing the data. Important relationships might be

embedded in complex sentence structures, making it hard to extract specific entity interactions without advanced text processing. As a result, analyzing unstructured biomedical literature has traditionally required labor-intensive manual review by domain experts or limited rule-based systems.

Another challenge is the inherent variability and complexity of biomedical language across different contexts. Research articles are not written with standardized vocabulary usage; different authors may describe the same biological process or clinical concept in varied ways. Moreover, biomedical text reflects the complexity of biological systems with their multifaceted interactions. In contrast to controlled vocabularies like medical ontologies, **biomedical literature presents irregular, multifaceted data**: biological entities often have multiple functions and interactions, and publications may discuss both established and novel findings. The technical style, specialized terminology, and the fact that findings are presented through different research perspectives make consistent information extraction problematic. These factors can confound algorithms attempting to identify entities (such as genes, diseases, or chemical compounds) and their relationships from raw text.

In addition, **scale and integration issues** pose further hurdles. The biomedical literature grows exponentially with thousands of new papers published daily (Kong, 2019), leading to information overload; processing this volume is complex and time-consuming. The heterogeneity of data sources (research articles, clinical notes, patents, databases, etc.) adds complexity in combining unstructured with structured biomedical data. Data quality issues like inconsistent entity mentions or missing context can propagate errors in analysis. All these challenges underscore why much of the unstructured biomedical text remains underutilized for systematic knowledge discovery. However, given the wealth of knowledge contained in biomedical literature, there is strong incentive to overcome these difficulties through advanced computational methods. In recent years, the field of **biomedical natural language processing (NLP)** has gained wide interest as researchers apply machine learning and language models to parse and derive meaning from biomedical text. These developments set the stage for transforming unstructured biomedical documents into structured forms that can drive scientific insights and clinical applications.

1.2 Research Motivation

1.2.1 The Need for Structured Information Extraction

Unlocking the value in unstructured biomedical text is a key motivation for this research. While researchers and clinicians rely on narrative documentation for communication and knowledge sharing, secondary use of this data (for systematic analysis, knowledge discovery, or automated reasoning) is limited by its unstructured nature. In order for biomedical information systems to fully utilize the vast literature and clinical documentation, the free-text portions must be converted into a structured, machine-readable form. **Information Extraction (IE)** is the crucial NLP task aimed at addressing this gap: it automatically identifies and encodes important pieces of information from unstructured text. By applying IE to biomedical literature, one can populate databases or knowledge bases with structured representations of scientific findings (e.g. identifying gene-disease associations mentioned in research abstracts and storing them as relationship triples in a knowledge graph).

There is a clear need for such **structured information extraction** to make use of the rich knowledge in biomedical texts. Studies have noted that much of the biomedical knowledge needed for drug discovery, clinical decision support, and research hypothesis generation is locked in free-text form (Liu et al., 2024). Automating the extraction of entities (genes, diseases, chemicals, species) and their relationships from biomedical literature would enable this knowledge to be integrated with existing structured databases and ontologies. This, in turn, supports advanced applications: for example, drug discovery systems could identify novel therapeutic targets from literature that might otherwise be overlooked, and researchers could systematically aggregate evidence from thousands of clinical notes and publications to generate new hypotheses. Indeed, combining structured and unstructured biomedical data has been shown to improve the performance of predictive models and knowledge discovery systems, since literature often contains context and novel findings not yet captured in databases.

In summary, the motivation is that **structured representation of information yields more actionable and computable knowledge**. Rather than leaving free-text biomedical data underutilized, converting it into structured form (through IE and encoding) can facilitate systematic analysis and knowledge discovery. This thesis is driven by the recognition that developing robust methods to extract structured knowledge from unstructured biomedical text will significantly enhance our ability to leverage scientific literature for research acceleration and clinical insights.

1.2.2 Graph Databases for Biomedical Knowledge

Once information is extracted from text, an appropriate data representation is needed to organize and utilize it effectively. **Knowledge graphs** (implemented via graph databases) have emerged as a powerful paradigm for representing complex interconnected information in biomedicine (Milvus, 2025). A knowledge graph consists of nodes (entities such as genes, diseases, chemical compounds, species, etc.) and edges (relationships between entities), forming a network of facts. This graph-based model aligns well with biomedical knowledge, which is inherently relational and heterogeneous. For instance, consider a disease that involves multiple genes, is treated by various drugs, and affects different biological pathways – a graph can naturally capture these many-to-many relationships (Gene→associated_with→Disease; Drug→treats→Disease; Disease→affects→Pathway; and so on) in a way that traditional relational databases struggle to represent without complex join tables.

Another key motivation for using knowledge graphs is their support for **knowledge discovery and hypothesis generation** in biomedical research. By organizing biomedical knowledge as a graph of entities and relationships, researchers can build systems that not only retrieve information but also discover novel connections through graph traversal and reasoning. For example, a knowledge graph might help researchers identify potential drug repurposing opportunities by finding paths connecting known drugs to diseases through shared molecular targets or biological pathways. Previous research has demonstrated that integrating biomedical literature into knowledge graphs enhances complex reasoning and allows researchers to query scientific knowledge in more meaningful ways (Rotmensch et al., 2017). Graph-based representations can unify knowledge from fragmented sources (different publications, databases, clinical records) and still preserve the connections needed for comprehensive biomedical understanding.

In summary, **graph databases provide the infrastructure to turn extracted biomedical facts into an "actionable" knowledge network**. By storing the output of information extraction in a knowledge graph, this research enables sophisticated queries, visualization of biomedical relationships, and improved knowledge discovery capabilities that leverage the full context of scientific literature and clinical data.

1.3 Research Question and Objectives

1.3.1 Primary Research Question

The primary question driving this thesis is:

How can unstructured biomedical text be transformed into an actionable knowledge graph using a multi-model approach for information extraction in healthcare and biomedical research?

We seek to determine how biomedical documents (e.g., research abstracts, clinical notes, scientific articles) can be automatically analyzed to extract structured knowledge and represented in a useful knowledge graph. This encompasses developing a multi-model pipeline (specialized models for entity recognition, linking, and relationship extraction) and evaluating its effectiveness. The focus is achieving a transformation that is both **accurate** and **actionable** for real-world use cases like knowledge discovery, literature analysis, and biomedical research acceleration.

1.3.2 Specific Objectives

To address the primary research question, the following specific objectives are defined:

1. **Design and implement a multi-model NLP pipeline for biomedical text.** This involves combining a named entity recognition (NER) model with entity linking and relationship extraction to process unstructured biomedical literature, identifying key biomedical entities (genes, diseases, chemicals, species) and their relationships from free text.

2. **Leverage both domain-specific and general language models for information extraction.** We will compare a medical domain-specific pretrained model (*MedGemma*) with a general-purpose model (*Gemma*) for extracting relationships, evaluating whether domain-specific fine-tuning provides superior accuracy in biomedical relationship extraction.

3. **Construct an actionable biomedical knowledge graph from extracted information.** Using identified entities and relations, build a knowledge graph in Neo4j by defining appropriate schema and automatically translating extraction outputs into Cypher queries that populate a structured, queryable representation of biomedical literature.

4. **Ensure compliance with ethical standards and data usage policies.** Implement appropriate data handling practices throughout the pipeline, ensuring proper citation and usage of public biomedical datasets while maintaining research integrity and following established guidelines for biomedical text mining.

5. **Evaluate the performance and utility of the proposed approach.** Develop evaluation frameworks measuring NER and relationship extraction accuracy

using precision, recall, and F1 metrics against gold-standard biomedical datasets, plus knowledge graph quality assessment. Compare domain-specific versus general models through statistical analysis to quantify their impact on extraction accuracy and graph completeness.

Through these objectives, the thesis systematically addresses the problem of converting unstructured biomedical text into a structured knowledge graph, from methodological development to evaluation of outcomes.

1.4 Ethical Considerations and Data Privacy

1.4.1 Responsible Use of Biomedical Data

Biomedical text data encompasses both publicly available scientific literature and sensitive clinical information, requiring careful attention to ethical standards and privacy protection. While research literature is generally publicly accessible, clinical text contains sensitive personal health information that demands rigorous privacy safeguards. For any clinical data processing, a fundamental step is **de-identification** to remove or obscure all identifiers that could link back to individual patients, following guidelines like the HIPAA Privacy Rule's Safe Harbor standard.

In this research, we primarily utilize publicly available biomedical datasets such as **BioRED** (Luo et al., 2022), which contains scientific abstracts from PubMed that do not contain personal health information. These datasets are obtained through legitimate research channels and used according to their intended research applications. However, our methodology is designed to be applicable to clinical data as well, with appropriate privacy protections. Any clinical text would be assumed to be properly de-identified prior to processing, with remaining institutional identifiers handled through filtering or tokenization to ensure no real patient or provider identities are exposed.

Data governance principles are followed throughout, limiting data use strictly to stated research objectives of developing and evaluating biomedical information extraction methods. We maintain research integrity by properly citing all data sources and following established protocols for biomedical text mining. The focus remains on extracting general biomedical knowledge patterns rather than identifying specific individuals or sensitive information. Because the thesis is conducted in English, we ensure all datasets contain English-language content suitable for our methodological approach.

1.5 Thesis Contributions

This thesis makes several contributions to the field of biomedical NLP and healthcare knowledge management:

- **Integrated Multi-Model Extraction Pipeline:** We develop a novel pipeline integrating multiple NLP models to transform unstructured biomedical text into structured knowledge graphs. The pipeline combines biomedical named entity recognition, entity linking to external knowledge bases, and relation extraction using large language models, demonstrating orchestrated end-to-end biomedical information extraction.

- **Domain-Specific vs. General Language Model Comparison:** The research provides comparative analysis of a domain-specific model (*MedGemma*) versus a general-purpose model (*Gemma*) for extracting relationships from biomedical literature. We quantify their strengths and weaknesses on relationship extraction accuracy and knowledge graph quality, offering insights into domain specialization value for biomedical and healthcare applications.
- **Construction of an Actionable Biomedical Knowledge Graph:** The thesis presents design and implementation of a biomedical knowledge graph capturing scientific literature in machine-readable form. We contribute methodology for translating raw biomedical text into graph database entries with domain-tailored schema, evaluating the graph's ability to answer complex biomedical queries and demonstrating practical utility for knowledge discovery.
- **Empirical Evaluation and Open Insights:** We conduct comprehensive experiments evaluating each system component and the final knowledge graph using established biomedical benchmarks like BioRED (Luo et al., 2022). The thesis reports detailed results, error analyses, and case studies illuminating common challenges in biomedical information extraction (Hier et al., 2025), contributing to broader understanding of effective approaches and remaining difficulties in transforming biomedical literature into structured knowledge.

Through the above contributions, the thesis advances both the methodology for biomedical information extraction and the practical considerations for deploying such techniques in real biomedical research and healthcare contexts.

Chapter 2

Literature Review and Theoretical Background

2.1 Natural Language Processing in Biomedicine

2.1.1 Evolution of Biomedical NLP

Natural language processing (NLP) techniques have been applied to biomedical text for several decades, encompassing both clinical documentation and scientific literature (Liu et al., 2024). Early biomedical NLP systems relied on **rule-based** or **dictionary-driven** approaches using handcrafted rules and medical term lookup lists. These systems, including tools like cTAKES, MetaMap, and PubTator, achieved high precision on well-defined patterns but were brittle with limited coverage and labor-intensive maintenance for evolving biomedical language across different domains (clinical notes, research abstracts, experimental protocols).

By the late 1990s and 2000s, **statistical machine learning** approaches emerged in biomedical NLP. Researchers formulated problems like entity recognition as supervised learning tasks using classification algorithms (SVMs, Maximum Entropy) trained on annotated biomedical corpora from both clinical and research domains. Sequence labeling methods like Hidden Markov Models and Conditional Random Fields became popular for **biomedical named entity recognition**, using probabilities rather than fixed rules but still requiring careful feature engineering by experts (Yazdani, Stepanov, and Teodoro, 2025).

Over the last decade, the field has undergone a **deep learning revolution**. Neural network–based models automatically learn features from large text amounts, achieving state-of-the-art results. Initial approaches used **recurrent** and **convolutional neural networks**, while **word embeddings** boosted performance by leveraging semantic similarities. The **transformer architecture** and BERT-like models brought another leap, enabling fine-tuning on biomedical datasets with excellent results across clinical notes, research abstracts, and experimental reports. This evolution progressed from rigid rule-based systems to statistical ML and now end-to-end deep learning, dramatically improving biomedical text processing despite remaining challenges.

2.1.2 Current State-of-the-Art Approaches

State-of-the-art (SOTA) NLP in biomedicine today is dominated by **advanced neural models**, especially transformer-based language models (Neumann et al., 2019). Researchers routinely use **domain-specific pretrained models** like **BioBERT**, **ClinicalBERT**, and **SciBERT**, which continue pretraining BERT on biomedical literature or clinical corpora. These models learn medical terminology nuances and significantly

outperform generic NLP models on tasks like biomedical NER, relation extraction, and question answering across both clinical and research text, making biomedical transformers the de facto starting point for biomedical NLP studies.

In parallel, **large language models (LLMs)** like GPT-3, GPT-4, and PaLM are beginning to influence biomedical NLP. These models demonstrate impressive capabilities in **zero-shot or few-shot settings**, such as parsing research abstracts, extracting gene-disease associations, or classifying drug-adverse effect relationships through prompting alone. While not specialized to biomedical text, their scale often compensates, and it is now state-of-the-art to explore *hybrid approaches* using domain-specific models for certain tasks and general LLMs for others (Liu et al., 2024).

Despite excitement around transformers and LLMs, practical state-of-the-art systems combine multiple components. Top-performing pipelines might use **biomedical NER models**, **rule-based methods** for specific patterns, and **transformer-based relation extractors**, often integrating with biomedical knowledge bases like UMLS. Current best approaches leverage (1) pretrained biomedical models, (2) large general models for zero-shot capabilities, and (3) domain knowledge to handle biomedical data idiosyncrasies across clinical and research contexts. This thesis builds upon these trends, employing specialized models and knowledge resources to maximize biomedical text extraction performance.

2.2 Named Entity Recognition in Biomedical Text

2.2.1 Traditional NER Methods

Named Entity Recognition (NER) in biomedical text involves identifying spans corresponding to biomedical concepts like genes, diseases, chemical compounds, species, or clinical procedures. Traditional approaches were dominated by **rule-based and dictionary-based methods**. Dictionary-based methods compile extensive entity lists for each category and scan text for exact or fuzzy matches across both clinical notes and research literature. Rule-based methods rely on human-crafted linguistic patterns, such as "if 'associated with' appears, following words until punctuation are likely Disease entities." Tools like **cTAKES**, **MetaMap**, and **PubTator** exemplify these approaches, using UMLS dictionary lookups and lexical variation generation for different biomedical text types (Yazdani, Stepanov, and Teodoro, 2025).

These traditional methods achieved **high precision** for entities matching their lists or patterns but showed **limited recall and poor adaptability**. New terminology or unexpected phrasing would be missed, and rule-based systems struggled with biomedical language variability across different text types (clinical notes vs. research abstracts). Maintaining these systems was difficult due to evolving medical vocabulary, varying institutional language use, and differences between clinical and research terminology. **Ambiguity** posed another challenge – terms like "cold" could refer to common cold (disease) or cold temperature (symptom) without complex understanding.

Traditional NER methods remain valuable in constrained scenarios due to their **interpretability** and quick deployment with decent lexicons. They complement modern methods as fallback components, using precision to validate neural model outputs. However, the field has largely moved beyond purely rule-based NER due to labor requirements and the need for higher recall and robustness.

2.2.2 Deep Learning Approaches for Biomedical NER

Deep learning approaches revolutionized biomedical NER by removing manual feature engineering and vastly improving generalization. Around the mid-2010s, researchers began applying **neural network models** like **BiLSTM-CRF** to biomedical NER tasks across clinical and research text. These models capture bidirectional context through LSTM and ensure consistent label sequences via CRF layers, quickly outperforming earlier hand-crafted feature models. Studies found that incorporating **character-level CNN or LSTM** sub-networks further improved recognition of rare or misspelled entities by capturing morphological patterns common in biomedical terminology.

A key advantage of deep learning NER is **word embeddings**. Biomedical terms like "aspirin" and "ibuprofen" have similar vector representations when trained on biomedical data, capturing their shared medication category. **Domain-specific embeddings** from biomedical corpora (e.g., PubMed articles, clinical notes) outperformed general embeddings, allowing models to leverage vast unlabeled text and recognize similar entities even when absent from training data.

By the late 2010s, **transformer-based models** took center stage. **BERT** introduced powerful contextual representation, with researchers fine-tuning pretrained BERT variants like BioBERT on labeled biomedical NER datasets. This approach achieves state-of-the-art performance, with transformers substantially outperforming earlier systems on benchmarks like the i2b2/VA challenge and BioCreative tasks. Deep models excel at handling **varied phrasing and contexts**, learning entity identification from context rather than explicit formatting across different biomedical text types.

Modern NER trends include **multiple corpora and multi-task learning** for improved generalization. Systems like **HunFlair** combine diverse biomedical datasets to create robust models that don't overfit to single text styles (Sänger et al., 2024). *Zero-shot* approaches like **GLiNER** (Yazdani, Stepanov, and Teodoro, 2025) attempt entity recognition without dataset-specific training, though they typically trail well-trained supervised models in accuracy.

Deep learning has become the dominant biomedical NER paradigm due to superior performance. The combination of powerful architectures, unsupervised pretraining, and larger datasets has pushed concept extraction to new heights. While challenges remain regarding labeled datasets and edge case handling, the gap between human and automated NER performance has narrowed significantly in the biomedical domain.

2.2.3 Biomedical Language Models

A major factor in modern biomedical NLP success is **biomedical language models** - large neural networks pretrained on biomedical text then fine-tuned for specific tasks. Notable examples include **BioBERT**, **ClinicalBERT**, **SciBERT**, and **PubMedBERT**. While general models like BERT learn from Wikipedia, they may not understand specialized biomedical vocabulary spanning clinical, research, and pharmaceutical domains. Domain pretraining allows models to pick up medical terminology - BioBERT trained on 4.5 billion PubMed words (Lee et al., 2019), while ClinicalBERT used EHR notes to learn clinical shorthand (Huang, Altosaar, and Ranganath, 2020), and SciBERT combined both research abstracts and clinical text (Beltagy, Lo, and Cohan, 2019).

These biomedical models have unequivocally improved biomedical NLP performance, serving as **foundational models** requiring little fine-tuning data. BioBERT-based NER models significantly outperform vanilla BERT on biomedical tasks because BioBERT "knows" terms like "bradycardia" and "BRCA1" from pretraining. Studies show BioBERT **"largely outperforms BERT and previous state-of-the-art models"** on biomedical benchmarks like JNLPBA, ChemProt, and BioRED. These models often include domain-specific tokenizers for better scientific term representation across different biomedical text types.

Another development is **continual pretraining and task-specific pretraining**, with models like **BioMegatron** and **PubMedGPT** (Bolton et al., 2024) trained from scratch on biomedical text. The general finding is that **domain knowledge leads to better understanding** - models reading research papers better recognize "PD-1 inhibitor" as a drug or "p53 mutation" as a genetic variant compared to general models.

Biomedical language models have limitations: high computational requirements and struggles with rare or new terms. However, they represent the state-of-the-art starting point. This thesis leverages such models expecting enhanced performance, with our **medical-specific versus general model** comparison essentially evaluating domain pretraining value - a topic of great current research interest.

2.3 Entity Linking Techniques

2.3.1 Knowledge Base Linking Methods

Once named entities are recognized, the next step is **entity linking** (entity normalization or grounding) - mapping text spans (e.g., "aspirin") to canonical identifiers in a knowledge base. In medical domains, this is crucial for resolving synonyms and variations to standard references, such as linking "heart attack" and "myocardial infarction" to the same concept.

A variety of techniques have been developed for entity linking in biomedicine, ranging from simple string matching to complex machine learning models:

- **String Matching and Dictionary Lookup:** The most straightforward method uses dictionaries of concept names from knowledge bases to match entity mentions. **Traditional techniques performed recognition and linking in one step**, scanning text for substrings appearing in the KB. Tools like **MetaMap** and **cTAKES** allow fuzziness (ignoring case, typos, word order) and rules for common variations (Yazdani, Stepanov, and Teodoro, 2025). These precision-oriented methods may miss entities without exact matches and struggle with ambiguous mentions having multiple possible mappings.

- **Rule-based and Heuristic Linking:** Some systems add heuristic rules beyond raw dictionary lookup, such as preferring matches whose semantic type fits context (mapping "aspirin" in medication sections to drug concepts). **Co-occurrence** heuristics use literature patterns to influence selection.

- **Machine Learning-based Linking:** Researchers introduced ML models for linking, formulating it as a **ranking problem** given candidate concept IDs. **DNorm** (Leaman, Dogan, and Lu, 2013) used pairwise learning-to-rank for disease normalization, while **TaggerOne** (Leaman and Lu, 2016) employed semi-Markov models for joint recognition and normalization. These approaches

require training data but outperform dictionary lookups by learning context-based disambiguation.

- **Neural Embedding-based Linking:** Deep learning approaches embed mentions and KB terms into vector spaces for similarity computation. The idea uses neural encoders to convert mentions and concepts into vectors where correct concepts are closest to mentions. Researchers have used **character-level BiLSTMs** or CNNs, like Phan, Sun, and Tay, 2019 who trained char-BiLSTMs with multiple objectives for mention-concept similarity and context coherence.

- **Pretrained Language Model for Linking:** Latest methods use transformers for entity linking. **BioSyn** (Sung et al., 2020) uses BioBERT to encode mentions and concept names, fine-tuning so mention vectors are close to true concept names and far from others. At inference, it performs fast nearest-neighbor search among concept embeddings, achieving high accuracy by capturing rich synonym and context understanding.

Each of these methods has trade-offs. Simpler methods (string match) are fast and don't require training data, but can't resolve ambiguity well. Advanced methods (neural linking) are more accurate in principle but require significant computation and annotated training data. Moreover, some neural methods struggle with the **scale** of biomedical KBs performing a vector search over millions of concepts can be slow or memory-intensive, which is why approximate search or pruning strategies are used. In practice, many systems adopt a **hybrid approach**: e.g., use quick string matching to get candidates, then a learned model to pick the best, combining speed with accuracy. Our work specifically leverages **SciSpacy's** linking (described below) for candidate generation, which is a strong unsupervised method, and we incorporate confidence scoring and domain rules to refine the results.

2.3.2 UMLS and MeSH Knowledge Bases

In the biomedical domain, the two predominant knowledge bases for entity linking are **UMLS** and **MeSH**. Understanding their nature explains why they're favored over general-purpose KBs like Wikipedia for clinical NLP.

The **Unified Medical Language System (UMLS)** (U.S. National Library of Medicine, 2024) is a comprehensive metathesaurus maintained by the U.S. National Library of Medicine, aggregating over 150 biomedical vocabularies into a unified framework. UMLS provides concept unique identifiers (CUIs) linking concepts to all names, synonyms, and semantic types. For example, aspirin has CUI C0004057 connecting "Aspirin," "Acetylsalicylic Acid," and brand names as a pharmacologic substance. UMLS is **massive** with over **3 million concepts** from sources like SNOMED CT, RxNorm, and ICD, enabling integration with electronic health records using standard codes.

MeSH (Medical Subject Headings) (Neumann et al., 2019) is a curated hierarchical taxonomy primarily for indexing PubMed articles, with about **30,000 main headings** covering diseases, chemicals, and anatomical terms. MeSH terms help librarians tag articles for retrieval and can serve as linking targets for biomedical literature. SciSpacy supports MeSH linking as a UMLS alternative. MeSH is **smaller and more curated** with well-defined concepts but won't include every clinical term or abbreviation.

Using medical-specific KBs offer clear advantages over general-purpose knowledge bases.

- **Coverage**: UMLS likely contains obscure medical acronyms that Wikipedia lacks.
- **Terminology consistency**: UMLS provides unified identifiers for concepts with different names across contexts.
- **Domain relevance**: UMLS/MeSH have medical-tailored semantic types and treatment relationships, enabling richer graph construction compared to general KBs with unrelated entries.

However, UMLS's comprehensiveness introduces ambiguity with overlapping concepts and names, making linking challenging. UMLS requires licensing (free for research), explaining why specialized tools pre-package the data. Our project predominantly uses **UMLS** for its broad coverage, occasionally referencing **MeSH** for alternate approaches.

2.3.3 SciSpacy's TF-IDF Character N-gram Matching

SciSpacy (Neumann et al., 2019) is an open-source library by the Allen Institute for AI that extends spaCy to scientific and biomedical text. It includes pre-trained biomedical NER models and an **EntityLinker** component for linking text spans to knowledge base entries. Our system leverages SciSpacy for entity linking.

SciSpacy's EntityLinker uses a **sparse vector model based on TF-IDF over character n-grams**. Each concept name in the knowledge base is indexed by character 3-grams (e.g., "aspirin" becomes "asp", "spi", "pir", "iri", "rin"). When linking entity mentions, SciSpacy computes TF-IDF weighted n-gram vectors and performs approximate nearest neighbor search to find closest matches by cosine similarity. This robust string matching handles minor spelling differences and partial matches, similar to tools like **QuickUMLS**.

The SciSpacy linker returns candidate KB identifiers with **similarity scores** for each entity span, typically the top 5 candidates sorted by TF-IDF similarity. For "COPD," it might return "Chronic Obstructive Pulmonary Disease" (score 0.95) and "Compulsive Obsessive Personality Disorder" (score 0.60). Users can set thresholds and retrieve concept names, definitions, etc., for each CUI.

SciSpacy integrates **abbreviation detection** using the **AbbreviationDetector** based on Schwartz & Hearst algorithm. It scans for patterns like "Full Form (ABBR)" and learns abbreviation definitions. With `resolve_abbreviations=True`, the EntityLinker attempts to link the *long form* rather than short form, greatly improving linking accuracy for abbreviations and acronyms.

SciSpacy's TF-IDF n-gram approach is *fast* and requires no training data, applicable to large KBs through efficient search libraries. However, it purely matches surface forms without using context beyond the mention string. We leverage SciSpacy to get candidate UMLS CUIs, then apply confidence thresholds and custom logic to finalize linking. It provides a solid baseline capturing most cases, especially when mentions closely resemble canonical names.

2.3.4 Challenges in Medical Entity Linking

Linking clinical entities to a knowledge base is a challenging task due to several inherent issues in medical text and the nature of biomedical knowledge bases:

- **Synonymy and Term Variation:** Medical concepts have numerous expressions - "heart attack," "myocardial infarction," "AMI," and "cardiac infarct" all refer

to the same concept. Linking systems must recognize these variants and unify them. UMLS provides synonyms, but text might contain unlisted variations like misspellings or specific descriptions. Ensuring different surface forms map to one concept rather than related but different concepts is a **recall** challenge.

- **Ambiguity (Polysemy):** Many medical terms are ambiguous - "RA" could mean rheumatoid arthritis, right atrium, or renal artery depending on context. When mentions map to multiple KB entries, linkers must pick correctly. This **precision** challenge requires using context (neighboring words, document metadata), which simple string-based methods don't fully exploit. Context-aware linking remains challenging, especially with limited training data.

- **Data Scarcity for Training:** Unlike NER, linking requires **mapping to concepts**, a complex annotation task needing domain experts. Creating gold-standard corpora where every mention maps to concepts like CUIs is time-consuming. Limited large training corpora means supervised ML approaches can overfit. This explains why dictionary and heuristic approaches remain popular - they don't need training data.

- **Knowledge Base Coverage and Granularity:** UMLS is huge but imperfect - concepts might be **missing** or at wrong granularity. For "brittle diabetes" (unstable Type 1), should it map to Type 1 diabetes or general diabetes mellitus? UMLS merges many sources, creating duplicative concepts that confuse linking systems. Choosing appropriate granularity and handling similar concepts requires nuanced understanding.

- **Abbreviations and Shorthand:** Abbreviations are extremely common but detection isn't guaranteed. Some are ambiguous or user-specific - "pt c/o CP" means "patient complains of chest pain." Linking "CP" requires knowing both the expansion and that chest pain is a symptom concept. Multi-step approaches (detect, expand, link) are needed, but failed expansion makes direct linking low-confidence.

- **Context and Relation Constraints:** Linking depends on context - "family history of breast cancer" should link to breast cancer *flagged as family history*, while "denies chest pain" involves negated context. While linking typically ignores negation, some applications want to encode that patients *don't have* problems. This challenge is usually left to pre-processing or post-processing rather than linking algorithms.

- **Scalability and Performance:** High-end methods like BERT encoding every mention against millions of concepts would be extremely slow. Scalability issues mean simpler methods are chosen as compromises. Even approximate nearest neighbor methods require high memory for concept embeddings. Making linking both *fast and accurate* is an active engineering problem - we chose SciSpacy's efficient approach to avoid bottlenecks.

Due to these challenges, **no single linking method is perfect**, and mistakes can propagate to later stages (e.g., linking "Cold" to "Common Cold" when meaning "cold sensation" creates erroneous knowledge graph entries). Our thesis addresses these through confidence scoring, abbreviation handling, and constraining entity types. Many state-of-the-art tools still include **manual or rule-based components**

for certain entities, acknowledging that combined approaches (simple methods for easy cases, complex methods for hard cases) are often the best practical solution – a philosophy we carry into our multi-model system.

2.4 Relationship Extraction and Knowledge Graphs

2.4.1 Relationship Extraction Methods

Identifying relationships between medical entities is the step that transforms isolated pieces of information into structured knowledge. In text, a **relationship extraction (RE)** system might detect, for example, that a certain drug is indicated for a disease, or that one clinical finding is a symptom of a condition mentioned elsewhere in the text. Extracting such relations from unstructured text allows us to then build edges in a knowledge graph (connecting the nodes that were identified via NER and linking).

Rule-Based Approaches

Early approaches to relation extraction in clinical text were predominantly **rule-based or pattern-based**, relying on human experts to define relationship patterns. Simple rules might be: *if a medication and condition appear with "for" or "to treat" between them, infer a Treats relation*. Examples include "<Drug> for <Condition>" templates or using trigger verbs ("caused", "due to") and syntactic patterns like dependency parsing to identify relationships.

Rule-based RE offers **precision-oriented and interpretable** advantages - extracted relations can be traced to specific rules. In critical applications like adverse event identification, carefully crafted rules provide reliability. Rules easily incorporate domain knowledge, such as "if X is a Finding and Y is a Disease connected by 'of', then X is a symptom of Y."

However, **coverage is a major issue**. Language variability makes it impossible to enumerate all relationship expressions. Pattern-based methods struggle with **negation and temporal expressions** - "No evidence of metastasis to the lung" contains relation indicators but is negated. This leads to **low recall**, with methods capturing only subsets of true relations and missing unexpectedly phrased relationships (Klug et al., 2024).

Despite limitations, rule-based approaches remain useful in certain settings for flagging specific events due to easier maintenance and verification. They complement statistical methods by handling obvious cases with high precision before machine learning processes the rest. Rule-based RE methods provide important groundwork with high precision and transparency but require extensive knowledge engineering and don't scale to real-world text diversity.

Machine Learning Techniques

As annotated clinical text datasets became available through shared tasks like i2b2 challenges, **machine learning (ML)** approaches to relation extraction gained prominence. Rather than manually specifying linguistic patterns, these approaches train models on labeled examples where entities and their relations are annotated, then learn to predict relations in new text.

Feature-based ML: Early ML methods used classifiers like Support Vector Machines or Logistic Regression, requiring text conversion to feature vectors. Researchers

engineered features capturing signals like words between entities, entity order and types, syntactic parse features, and section information. For example, determining "adverse reaction" relations might use features like temporal proximity ("Penicillin allergy") or connecting verbs ("causes"). Feature-based ML was successful in competitions, with many top i2b2 entries using SVMs with cleverly designed features, substantially beating pattern-matching in recall while maintaining precision (Klug et al., 2024).

One limitation is these models **don't generalize well to contexts not represented in features**. If important cues aren't captured as features, models can't learn from them. They also required extensive domain-specific tweaking, essentially moving the burden from writing rules to writing good features.

Neural Network ML: Relation extraction moved toward neural approaches around mid-2010s. Models like **CNNs** and **RNNs** were applied to word sequences between target entities. Classic architectures represent sentences with marked entity positions as embedding sequences, feed into networks, and output relation classes. More recently, transformer models like BERT achieve state-of-the-art results by inserting special tokens around entities and using the output for relation classification.

ML approaches (especially neural ones) greatly improved **recall and adaptability**, finding subtly expressed relations that rules might miss without requiring human pattern enumeration. However, deep models need substantial labeled data, challenging in clinical domains. Many models use combined datasets or synthetic data supplementation. **Error interpretability** remains difficult when learned models make mistakes.

The current trend favors neural RE when data is available, consistently outperforming feature-based methods. Hybrid systems combine neural outputs with rule-based frameworks for optimal results. In our project, we explore using large language models for **relationship extraction**, taking the neural approach to extremes by leveraging models trained on enormous general text corpora.

Large Language Models for Relationships

The latest frontier in relationship extraction uses **Large Language Models (LLMs)** like GPT-3, GPT-4, and PaLM. These models, trained on diverse internet-scale text, perform tasks via prompting without explicit task-specific training through general language understanding. In biomedical relation extraction, researchers investigate LLM capabilities for identifying relationships without training. Findings are promising: **"Large Language Models have demonstrated impressive performance in biomedical relation extraction, even in zero-shot scenarios."** Laskar et al., 2025.

There are a couple of ways LLMs are used for RE:

- **Prompt-based Classification:** You can present LLMs with prompts describing the relation extraction task and specific instance. For example, to classify gene-disease relations, prompt GPT-4 with: *"Determine if there is a positive, negative, or no relation between [BRCA1] and [breast cancer] in: 'The BRCA1 mutation is associated with higher breast cancer risk.' Answer '1' for positive, '0' for no relation."*. This turns RE into question-answer format. With carefully engineered prompts, LLMs achieve accuracy comparable to supervised models on benchmarks, requiring **no training phase** on task-specific data - extremely useful in clinical RE where annotated datasets are scarce.

- **Prompt-based Extraction (Generative):** Instead of classifying given options, LLMs can directly extract relations open-endedly. For instance: *"Extract all*

<drug, condition> treatment pairs from the text..." or ask for structured JSON output listing problems with associated tests/medications. This leverages LLMs' generative nature for flexible information extraction. Tools like GPT-4 can parse complex text and output triples with surprising accuracy, inherently performing joint NER and RE by identifying entities and relations simultaneously during generation.

Using LLMs for RE brings **new challenges**. First, evaluation is tricky since LLMs might express relationships differently from gold standards, making automatic evaluation difficult. Second, they can **hallucinate** - producing plausible but unstated relations, like "filling in" likely treatments not explicitly mentioned. Ensuring models stick to text evidence is active research (chain-of-thought prompting, instructions against unmentioned facts). Third, **privacy and deployment**: powerful LLMs accessible only via cloud APIs are problematic for patient data, spurring interest in smaller domain-specific models for local hospital deployment (Ji et al., 2023).

Despite issues, LLMs' zero-shot or few-shot RE ability is potentially game-changing, dramatically reducing development time and adapting to multiple tasks easily. Our thesis compares a **medical-specific model** (MedGemma3-4b) with a **general-purpose model** (Gemma3-4b), examining how much general LLMs know about clinical relationships versus medical-tuned models. Prior work suggests general models are strong but domain-tuned models maintain advantages in specialized accuracy. LLMs represent cutting-edge relationship extraction, offering flexibility and massive implicit knowledge, likely playing increasing roles in clinical information extraction systems.

2.4.2 Knowledge Graph Construction

Graph Database Technologies

A **knowledge graph** is essentially a network of entities (nodes) linked by relationships (edges) that represent facts or assertions. Once we extract entities and relationships from text, we need to store them in a structured way for querying and analysis. This is where **graph database technologies** come into play. Unlike traditional relational databases that use tables, graph databases are designed to store and query data in terms of nodes and edges, which is a natural fit for knowledge graphs.

There are two main paradigms for graph data management: **RDF triple stores** and **property graph databases**.

- **RDF Triple Stores:** These databases store data as triples (subject, predicate, object) following the Resource Description Framework standard. For example, (Aspirin, treats, Diabetes) would be a triple. Triple stores like **Apache Jena TDB**, **Virtuoso**, and **GraphDB** use SPARQL query language, which resembles SQL but is tailored to graph patterns with variables. RDF systems excel at enforcing schemas/ontologies and integrating data from multiple sources via shared identifiers, making them popular in academic knowledge graph work requiring semantic interoperability (Cowell and Smith, 2020).

- **Property Graph Databases:** More commonly used in industry, property graph models allow each **node** to have labels and properties (key-value pairs), while each **relationship** has a type and properties. For example, a "Medication" node might have `name="Aspirin"` and `dosage="325mg"`, connected via "TREATS" edge with `source="doctor_notes"` property. This flexible model allows attaching contextual data directly on nodes/edges. **Neo4j** leads this category

with user-friendly, optimized graph queries, alongside **Amazon Neptune**, **JanusGraph**, and **TigerGraph**.

Graph databases allow efficient traversal of the graph. For instance, a query like "find all medications that treat diseases which have symptom X" would involve traversing from a symptom node X, to disease nodes connected by "hasSymptom" edges, then from those diseases out via "treatedBy" or "treated_with" edges to medication nodes. Doing this with SQL would be complex and slow if the data is heavily relational and many joins are needed, whereas graph DBs are optimized for such multi-hop queries.

Moreover, graph databases align with how we conceptualize knowledge: they can directly store an edge saying **Drug→treats→Disease**, which is more intuitive than having to indirectly connect them via foreign keys in tables. In a knowledge graph, relationships are first-class citizens – you can query them, filter on them (e.g., find all "inhibits" relations between genes and proteins, etc.), and even put constraints on paths.

In summary, graph database technologies provide the backbone for **storing** and **querying** the knowledge graph that results from our NLP pipeline. They handle the heavy lifting of managing connections at scale. The choice between RDF vs property graph often comes down to use case; we have chosen a property graph approach (Neo4j) for its straightforwardness and the ability to easily attach attributes to nodes/edges which is useful for tracking metadata of extracted info.

Neo4j and Cypher Query Language

Neo4j (Neo4j, Inc., 2023) is a widely-used graph database that implements the property graph model and has become a standard for many knowledge graph projects. It's known for its performance, developer-friendly interface, and a rich ecosystem of tools. In the context of our thesis, Neo4j serves as the platform where the extracted knowledge graph is built and stored. Here's why we chose Neo4j and how we use it:

- **Data Model:** In Neo4j, data is stored as nodes and relationships with labels like `:Person`, `:Condition`, `:Drug` for nodes and relationship types like `DIAGNOSED_WITH`, `TREATS` for edges. Each can have properties – patient nodes link to condition nodes via `HAS_CONDITION` edges with `onset_date` or `status` properties. Our extracted graph has nodes representing medical concepts and edges representing text relations. Neo4j's property graph capability allows storing confidence scores or context as edge properties for later analysis.

- **Cypher Query Language:** Neo4j uses **Cypher**, a declarative language for querying graphs with pattern matching. For example:

  ```
  MATCH (d:Drug {name:"Aspirin"})-[:TREATS]->(c:Disease)
  RETURN c.name;
  ```

  This finds diseases treated by Aspirin using ASCII-art notation where `()` denote nodes and `-[]->` denote relationships. We generate Cypher programmatically, using `MERGE` to create nodes and relationships without duplicates. Cypher's readability makes graph construction verification easier.

- **Advantages of Neo4j:** Neo4j is optimized for **graph traversals** - queries requiring multiple JOINs in SQL are fast due to index-free adjacency where nodes directly know their connections. Features include full-text indexing and APOC library for graph algorithms. The web-based browser interface allows manual knowledge graph exploration with visual node and relationship plotting, revealing patterns or linking errors.

- **Cypher Query Generation:** Our methodology translates extracted information into Cypher queries for each processed clinical note. We batch queries and use transactions for efficient insertion, resulting in a Neo4j database containing the consolidated knowledge graph from all notes.

In summary, **Neo4j** provides a robust platform for our knowledge graph, and **Cypher** allows us to interact with that graph in a flexible way – whether to populate it or to retrieve insights. Many published clinical knowledge graph projects (and industry solutions) use Neo4j due to these strengths (Zimbres, 2024). By using Neo4j, we align with best practices and ensure that our graph is not just a conceptual outcome, but a queryable database that can answer complex questions about the data model we've constructed.

2.4.3 Biomedical Knowledge Graph Applications

Knowledge graphs (KGs) in the biomedical domain unlock a range of powerful applications by enabling connections across different types of medical data and knowledge. By converting unstructured biomedical text into a graph of entities and relationships, we create a structure that can be traversed and analyzed to support clinical care, biomedical research, and drug discovery. Here we review some key applications of biomedical knowledge graphs, as reported in literature and practice (Milvus, 2025):

- **Integrating Biomedical Data for Comprehensive Analysis:** Knowledge graphs break down data silos by representing biomedical entities (genes, diseases, drugs, species, pathways) as connected subgraphs. A KG might link gene nodes to disease and drug nodes, enabling complex queries like "Find genes associated with Disease X that are targets of Drug Y." This supports both clinical applications and research hypothesis generation. For clinical use, patient nodes connected to "Type 2 Diabetes" can link to genetic risk factors and guideline nodes. For research, disease-gene-drug connections enable systematic literature analysis and drug repurposing opportunities.

- **Knowledge Discovery and Hypothesis Generation:** Knowledge graphs enable systematic exploration of biomedical relationships extracted from literature. KGs can reveal indirect connections like "Gene A $\rightarrow$ Disease B $\rightarrow$ Drug C" suggesting potential therapeutic targets. Research applications include identifying novel drug-disease associations, understanding disease mechanisms through pathway analysis, and discovering biomarkers through entity relationship traversal. Graph queries implement reasoning by pattern matching, supporting both clinical decisions and research hypotheses with comprehensive relationship consideration beyond isolated findings.

- **Clinical Decision Support (CDS):** Knowledge graphs enhance clinical decision support by encoding medical knowledge and enabling reasoning. KGs can model clinical rules like "Patients with Condition X should receive Test

Y every 6 months" as relationships between condition and procedure nodes. Drug→condition→contraindication graphs enable CDS systems to detect conflicts among patient medications. Research prototypes use KGs for diagnosis suggestions by connecting patient symptoms to disease nodes through integrated clinical and research knowledge graphs.

2.5 Related Work Summary and Research Gap

In this chapter, we reviewed the literature across the spectrum of our task: from natural language processing in the biomedical domain (entity recognition and linking) to relationship extraction and knowledge graph construction. The existing body of work demonstrates substantial progress in each of these individual areas:

- **Biomedical NLP Maturity:** The evolution from rule-based systems to statistical ML and now deep learning/large language models has greatly enhanced biomedical text extraction accuracy across clinical notes and research literature. Robust tools exist for biomedical NER (BiLSTM-CRF, BioBERT-based taggers) and relation extraction, with transformers and prompt-based LLM methods leading benchmarks. Literature consistently shows domain-specific knowledge (biomedical embeddings, clinical and research corpora training) yields better results than general models, with domain-tuned models like BioBERT and SciBERT outperforming general BERT on biomedical tasks.

- **Entity Linking and Knowledge Bases:** Numerous approaches exist for linking entities to biomedical KBs, from dictionary methods (MetaMap, PubTator) to advanced neural algorithms. UMLS and MeSH provide essential medical ontology backbones for standardization across clinical and research domains. Despite sophisticated linking models, many practical systems still incorporate simple methods for efficiency and completeness - even state-of-the-art systems like BERN2 and PubTator use hybrid approaches with dictionary lookups. This informs our approach to use proven tools (SciSpacy) with custom logic rather than training new linkers.

- **Knowledge Graph Construction & Use:** Multiple reports confirm biomedical knowledge graph feasibility and value, including drug interaction KGs from literature, gene-disease KGs from research papers, and integrated clinical KGs for outcome prediction (Rotmensch et al., 2017). Combining diverse data sources (research literature, clinical notes, structured databases, guidelines) enables insights impossible with siloed data. However, existing works focus on specific domains rather than comprehensive solutions. Our thesis carves a niche: generating knowledge graphs from unstructured biomedical text with **multi-model pipelines** comparing general versus specialized models.

After surveying the literature, we identify two key gaps that motivate our research:

1. **Lack of a Unified Multi-Model Pipeline Evaluation:** Many studies address individual components (NER, linking, RE, or KG construction) in isolation rather than end-to-end systems. There is a need for exploring **combined approaches** where each task uses the best-suited model, evaluating the complete pipeline on real biomedical text. Our thesis addresses this by integrating domain-specific NER (GLiNER),

proven linking (SciSpacy/UMLS), and large language models for relation extraction. We explicitly combine specialist tools and analyze how they complement or hinder each other, including error propagation and parallel processing impacts. Our work aims to provide a blueprint for multi-model information extraction pipelines in biomedicine - an area not comprehensively covered in literature.

2. **Limited Comparative Analysis of General vs. Domain-Specific Models:** Another gap is understanding trade-offs between general and domain-specific models in biomedical IE tasks. While domain-tuned models typically outperform on individual tasks, very large general models (e.g., GPT-4) show surprisingly strong biomedical capabilities. Could appropriately prompted general models perform as well as domain-specific models on biomedical relation extraction? Literature lacks direct head-to-head comparisons in full information extraction contexts. We fill this gap by comparing "MedGemma3-4b" (Google's medical-finetuned model) versus "Gemma3-4b" (an instruction tuned version of the base model) on relationship extraction for knowledge graph construction. This study examines whether medical models better capture biomedical context or if general models compensate with parametric knowledge, and identifies relation types where each excels - providing systematic evaluation within one pipeline.

By addressing the above gaps, the thesis aims to advance the state-of-the-art in transforming unstructured biomedical text into actionable knowledge graphs. We not only propose a novel pipeline architecture but also critically evaluate the components and alternatives. The end result will be a clearer understanding of how to best combine tools and models for biomedical information extraction, and evidence-based insight into the value of domain-specific versus general AI models in this setting. This bridges the isolated advances in literature into an integrated solution, moving a step closer to practical deployment in biomedical research and healthcare environments.

Chapter 3

Methodology

This chapter details the multi-model methodology for transforming unstructured biomedical text into a structured medical knowledge graph. The overall approach is a **pipeline** that sequentially performs entity recognition, entity linking, relationship extraction, and graph construction. By chaining specialized components, the system produces an **interpretable representation of medical concepts (e.g. drugs, diseases) and the relations among them**, enabling integration of context and supporting clinical insights. Key design decisions, data preprocessing steps, model configurations, and implementation optimizations are discussed in the following sections.

3.1 System Architecture and Pipeline

3.1.1 Overall System Design

The proposed system follows a modular pipeline architecture, where each stage transforms the biomedical text and feeds into the next stage. The architecture consists of the following major components and data flow:

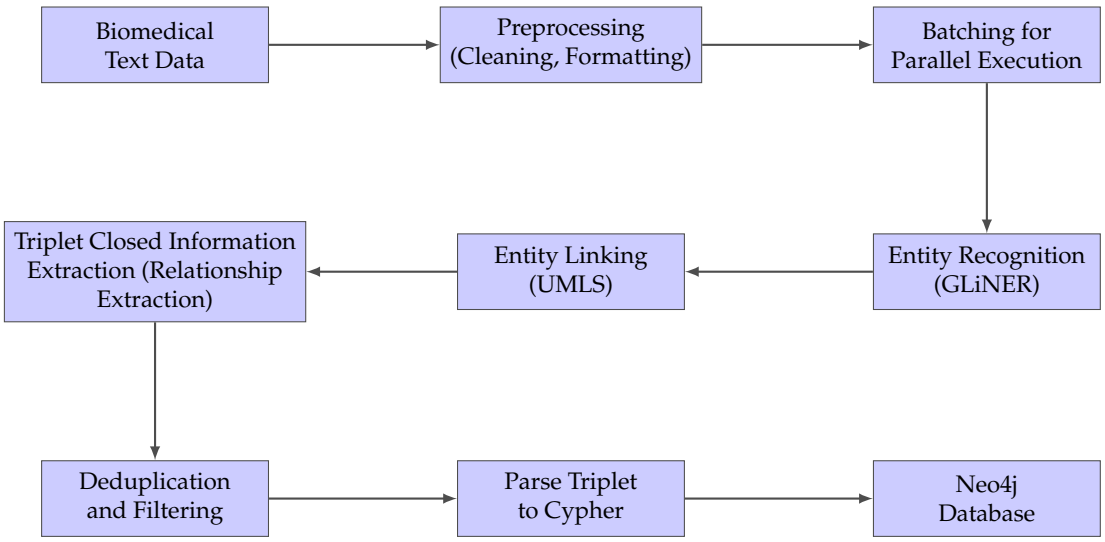

FIGURE 3.1: System architecture pipeline showing the transformation of biomedical text into a knowledge graph

1. **Input and Preprocessing:** Raw biomedical text (free-text) is first collected and preprocessed (cleaning, formatting) to ensure consistent input for NLP tasks.

2. **Entity Recognition:** A biomedical Named Entity Recognition (NER) model identifies spans of text corresponding to biomedical concepts (e.g. genes, diseases, chemical compounds, variants, species). We employ the GLiNER model for this step (see Section 3.3.1).

3. **Entity Linking:** Detected entity mentions are normalized and linked to a canonical identifier in a medical knowledge base. We use SciSpacy's UMLS linker to map each mention to a UMLS Concept Unique Identifier (CUI), providing standardized meanings (Section 3.3.2).

4. **Relationship Extraction:** A large language model (LLM) processes the linked entity list and full text to extract relationships between the medical entities. The model outputs structured triple relationship assertions (Section 3.4).

5. **Knowledge Graph Construction:** The extracted entities and relationships are inserted into a Neo4j graph database. Nodes represent medical concepts (annotated with UMLS CUIs and types) and edges represent the relationships between concepts. Cypher queries are generated to create or merge these graph elements (Section 3.5).

6. **Post-processing & Optimization:** Throughout the pipeline, confidence filters, validation checks, and parallelization are applied to ensure quality and efficiency (Section 3.6).

Each component addresses a specific task, and their integration yields a coherent end-to-end system. This design allows independent optimization of each module and the ability to swap models if needed. The multi-step **"recognition→linking→relation extraction→graph loading"** strategy aligns with recommended knowledge graph construction practices, ensuring that unstructured text is incrementally converted into a structured, interoperable knowledge graph (Zimbres, 2024).

3.1.2 Component Integration Flow

The flow of data between components is carefully orchestrated to preserve context and accuracy. The pipeline is implemented as a *sequential workflow* where each module consumes the outputs of previous steps:

- The **Preprocessing** stage standardizes the input. This may include lowercasing (if required by the NER model), removing extraneous whitespace or headers, and ensuring patient-identifying information is excluded for privacy (see Section 3.2).

- The **NER component** (GLiNER) is applied next on the cleaned text. It produces a set of entity mentions (text spans) each with an entity type (e.g. *Condition*, *Drug*, *Test*). These mentions are passed forward as Python objects (e.g. spaCy `Span` objects) attached to the document representation.

- The **Entity Linking** module (SciSpacy's `EntityLinker`) takes each NER-detected span and searches for candidate concepts in the UMLS knowledge base. Because this linker is integrated as a spaCy pipeline component, it can enrich the detected `Span` with a list of candidate CUIs and similarity scores (available via `Span._.kb_ents`). We configure the linker to resolve abbreviations and to attach the top-ranked CUI for each mention above a confidence threshold (details in Section 3.3.3). The result is that each entity mention from NER

is now linked to a unique medical concept ID or marked as *unlinkable* if no high-confidence match is found.

- For **Relationship Extraction**, the entire biomedical document along with the recognized entities is provided to the LLM. In practice, we construct a prompt that includes the text and possibly the list of identified entities, asking the model to output relationships (e.g. cause/effect, treatment, associations) among those entities. The integration is such that the model's output can be traced back to the original entities by name or CUI. This output is captured as structured relation triples.

- Finally, the **Graph Construction** module translates the entities and relations into Cypher queries. The integration here uses the results of previous steps (entity CUIs, names, types, and relation types) to form `MERGE` statements. For example, if "C0011849" (Diabetes mellitus) and "C0027796" (Neuropathy) are two CUIs with a *causes* relationship, the pipeline will generate a Cypher command to `MERGE` nodes for each (with labels/types) and a `[:CAUSES]` relationship between them. These queries are executed on the Neo4j database to update the knowledge graph.

Throughout this flow, **state is maintained** so that each mention's context is known. For instance, if an LLM-produced relationship mentions an entity not recognized by NER, we detect that during parsing and handle it (either by discarding that relation or attempting to link the new entity, as discussed in Section 3.4.2). This ensures consistency: only vetted entities make it into the graph. The sequential integration also means errors can propagate (e.g. a linking error could affect relation extraction), so each component is tuned to maximize precision, and wherever possible, the pipeline includes verification steps (for example, ensuring that relation arguments have valid CUIs). The design balances modularity with tight integration, creating a robust system to extract structured knowledge from biomedical text.

3.2 Data and Preprocessing

3.2.1 Biomedical Literature Dataset and Preparation

Our dataset consists of biomedical literature from the **BioRED** dataset (Luo et al., 2022), a comprehensive corpus for biomedical relation extraction. BioRED contains manually annotated PubMed abstracts with multiple entity types (gene/gene products, diseases/phenotypic features, chemical entities, variants, species, and cell lines) and their relationships. The dataset provides 600 annotated documents with more than 20,000 entity mentions and 6,000 relationship annotations across eight relation types including Association, Positive_Correlation, Negative_Correlation, Bind, Comparison, Cotreatment, Drug_Interaction, and Conversion. These abstracts represent diverse biomedical research areas and contain domain-specific terminology, complex medical concepts, and scientific relationships. The dataset is particularly valuable for developing and evaluating biomedical NLP systems as it captures the complexity of scientific literature while providing gold-standard annotations. Before applying the NLP pipeline, we perform several preprocessing steps to prepare this data:

- **PubTator Format Parsing:** The BioRED dataset is provided in PubTator format, a standardized format for biomedical text annotations. Each document

contains a PubMed ID (PMID), title, abstract, entity annotations with character offsets, and relation annotations linking entity IDs. We parse this format to extract the text content and gold-standard annotations for evaluation purposes.

- **Text Reconstruction:** We combine the title and abstract sections to form complete documents for processing. The character offsets from entity annotations are preserved to enable precise evaluation against our pipeline's predictions. We ensure proper spacing between title and abstract to maintain text coherence.
- **Entity Type Mapping:** BioRED uses specific entity type labels (GeneOrGeneProduct, DiseaseOrPhenotypicFeature, ChemicalEntity, etc.) which we map to our pipeline's entity categories. This mapping ensures compatibility between the dataset's annotation schema and our NER model's expected input types.
- **Relation Type Standardization:** The dataset includes eight relation types with specific biomedical meanings. We preserve these relation types for evaluation while ensuring they align with our relationship extraction model's output format. Relations are provided as entity ID pairs, which we resolve to actual text mentions using the entity annotations.

After preprocessing, the biomedical abstracts are properly formatted and their annotations are structured for evaluation. This careful preparation facilitates accurate assessment of our pipeline's performance against gold-standard annotations. The preprocessing ensures that entity boundaries are preserved, relation pairs are correctly mapped, and the text maintains its scientific accuracy. These steps address the **"garbage in, garbage out"** concern - proper data preparation is essential for meaningful evaluation and reliable knowledge extraction from biomedical literature.

3.3 Entity Recognition and Linking

3.3.1 GLiNER Model for Medical Entity Recognition

For identifying medical entities in text, we utilize the **GLiNER** model – a *Generalist and Lightweight Named Entity Recognition* framework adapted for biomedical data. Traditional medical NER is challenging due to vast evolving healthcare vocabulary and limited labeled training data. GLiNER offers **open-domain, zero-shot NER** using *natural language descriptors* rather than fixed entity taxonomies. We can prompt GLiNER with definitions like medical condition" or drug name," and it identifies corresponding text spans.

GLiNER-BioMed leverages large language models for annotation, then distills knowledge into smaller, efficient NER models. Developers generated synthetic training data using LLMs, then trained GLiNER models on this data. The result achieves state-of-the-art biomedical NER performance with 6% F1-score improvement over previous systems in zero-shot scenarios, statistically significant at $p < 0.001$ (Yazdani, Stepanov, and Teodoro, 2025).

In our pipeline, we chose GLiNER for its generalization ability, configuring it with **medical entity type prompts**: *Problem/Condition*, *Treatment/Drug*, *Test/Procedure*,

and others as needed. GLiNER can tag entities without dataset-specific retraining. For *"The patient was started on metformin for diabetes mellitus"*, GLiNER identifies **"metformin"** as Drug and **"diabetes mellitus"** as Condition based on prompt understanding.

GLiNER's **lightweight architecture** is smaller and faster than full LLMs, suitable for scanning long documents. It effectively *distills* larger models' linguistic knowledge into specialized NER tasks, running efficiently on available compute (Apple Silicon CPU/GPU) with reasonable speed crucial for biomedical text processing.

3.3.2 UMLS Entity Linking with SciSpacy

Once entities are recognized, we perform **entity linking** to anchor mentions to standardized medical ontology using SciSpacy's UMLS **EntityLinker** for biomedical normalization. SciSpacy provides spaCy models tailored to scientific/clinical text (Neumann et al., 2019), with entity linker containing a built-in knowledge base from the **Unified Medical Language System (UMLS)** Metathesaurus integrating millions of biomedical concepts.

SciSpacy's linking algorithm uses **string similarity** with *character n-grams*, representing entity mentions as bag-of-character-trigrams with **TF-IDF weighted vectors**. Every UMLS concept name is indexed by character trigrams, enabling approximate nearest neighbor search for most similar concepts. This finds overlapping substrings - robust for biomedical text with lexical variants, like cardiac infarction" matching *Myocardial Infarction"* through trigram overlap.

We configured SciSpacy with `linker_name="umls"` and `resolve_abbreviations=True` to expand short forms before linking. The UMLS knowledge base includes primary vocabularies totaling roughly 3 million concepts. We kept the default **similarity threshold** of 0.7, meaning linkers only assign concepts when cosine similarity between TF-IDF trigram vectors is $\geq$ 0.7, filtering tenuous matches and improving precision.

During linking, SciSpacy produces candidate concept IDs with similarity scores for each entity. We select top-ranked candidates exceeding the threshold. For metformin," the linker returns (CUI: C0025598, Metformin", score 0.98), providing *canonical representation* with UMLS Concept Unique Identifiers and standardized names, definitions, and semantic types.

Linking to UMLS creates **interoperable knowledge graphs**. Grounding entities to UMLS CUIs aligns nodes with established ontology, facilitating healthcare data integration and hierarchical queries. It consolidates synonyms - heart attack" and myocardial infarction" link to the same CUI, creating single graph nodes and avoiding duplication (U.S. National Library of Medicine, 2024).

3.3.3 Confidence Scoring and Abbreviation Detection

In this stage, we refine the outputs of NER and linking by incorporating confidence measures and handling abbreviations explicitly:

- **Confidence Scoring:** Both NER and linker provide internal scores. After entity linking, we use SciSpacy's **similarity score** as confidence proxy. Poor matches (0.4 similarity) may be dropped, while high scores (close to 1.0) indicate confident matches. We examine score distributions - if best match is 0.72 and second is 0.71, that's ambiguous; if best is 0.85 and second is 0.60, that's clear-cut. We

set thresholds: below 0.7 marks entities as "Unlinked," above 0.9 accepts outright, and 0.7-0.9 accepts with lower confidence flags. This approach maintains knowledge graph quality by minimizing spurious nodes/edges.

- **Abbreviation Detection:** Biomedical literature contains numerous abbreviations and acronyms requiring resolution for correct linking. We integrate SciSpacy's **AbbreviationDetector** implementing Schwartz & Hearst algorithm (Schwartz and Hearst, 2003), scanning for patterns like "Full Form (Abbr)." For *"tumor necrosis factor alpha (TNF-α)"*, it detects **"TNF-α"** abbreviates **"tumor necrosis factor alpha."** With `resolve_abbreviations=True`, the EntityLinker uses long forms for UMLS matching, yielding correct high-confidence links. This particularly helps with common biomedical abbreviations like "IL" (interleukin) by using local context for expansion.

By applying confidence scoring and abbreviation resolution, we **improve the precision and recall** of the entity linking process. High-confidence links and fully expanded terms result in more correct nodes in the graph, and fewer missed entities. These measures also reduce noise for the next stage: the relationship extraction model will receive text where abbreviations are already expanded (in the `Doc` object's context) and where uncertain entities can be treated cautiously. Overall, this step solidifies the foundation of the knowledge graph by ensuring that we have *trusted, well-defined entities* to work with.

3.4 Relationship Extraction

3.4.1 Model Selection and Prompt Engineering

Extracting relationships from biomedical text involves understanding semantic connections between medical entities. We approach this with **Large Language Models (LLMs)** that interpret text meaning and generate relational triples. We consider two LLMs: **Gemma** (general-purpose model) and **MedGemma** (specialized variant fine-tuned for medical domains). MedGemma, developed by Google DeepMind in 2025, demonstrates advanced medical understanding and clinical reasoning capabilities, building on Gemma's strengths with domain-specific training (Google DeepMind, 2025; Google AI, 2025).

Given accurate relation extraction's critical nature, we experiment with both models. MedGemma (4B variant) is expected to excel at medical relationships (drug-disease interactions, symptom-disease associations), while Gemma provides a baseline for non-medical-tuned performance. The pipeline is *model-agnostic*, accommodating any LLM accepting prompts and returning text.

We use a **prompt-based approach** (in-context learning) rather than fine-tuning to leverage models directly. Prompts are carefully engineered for structured output, following best practices that clearly define tasks, provide examples, and indicate desired formats (Reynolds and McDonell, 2021).

- A brief task description: e.g., *"Extract all clinically relevant relationships between medical concepts in the following text."*
- A format instruction: e.g., *"Provide the relationships as a list of triples (Subject, Relation, Object) using the exact entity names from the text."* We explicitly ask for the model to use the entities as mentioned, to ease alignment with CUIs.

- Optionally, a few-shot example: for instance, showing the model an example sentence and the extracted triple from it. We might include one or two demonstration pairs if it improves performance, although with very large models often a clear instruction suffices.
- The context text: the actual biomedical document or a segment of it, possibly truncated to stay within token limits of the model.

An example prompt might be:

```
Extract all relationships between medical entities in the text.
Use the format (Entity1, Relation, Entity2).
Text: "The patient's diabetes caused peripheral neuropathy and
he was prescribed gabapentin for pain management."
Relationships:
1. (diabetes, causes, peripheral neuropathy)
2. (gabapentin, treats, pain)
```

In this prompt, we provided a made-up example demonstrating the expected output format. The actual note's text would follow after "Text:" and we would expect the model's completion to list similar triples.

We also incorporate prompt elements to handle nuances: e.g., instruct the model to ignore trivial relations or to only output relations that are explicitly or implicitly stated (to avoid hallucination). We emphasize that the output should not include any entity not found in the text. This is important because LLMs have a tendency to infer or hallucinate facts; by explicitly saying *"use only entities from the text"*, we reduce the chance the model introduces an unrelated concept.

To summarize, our prompt engineering strategy focuses on clarity, examples, and format enforcement. We aim to push the model to behave almost like a rule-based extractor but backed by its deep understanding of language. This harnesses the best of both worlds: the model's intelligence and a deterministic output scheme. We will quantitatively compare the two chosen models' outputs later, but here it's worth noting that using an LLM for relation extraction aligns with the latest research trends in knowledge graph construction. Large foundation models have been successfully used to perform relation extraction without extensive task-specific training, by virtue of their pre-trained knowledge and language understanding (Singhal et al., 2022). Our methodology capitalizes on this capability by employing prompt-based LLM queries to extract rich relational information from text that simpler models or rule-based systems might miss.

3.4.2 Output Parsing and Validation

After the LLM produces candidate relationships in text form, we need to parse these outputs and validate them before integration into the knowledge graph. This step is critical for maintaining accuracy and consistency, as the raw model output may contain noise or require interpretation.

- **Output Parsing:** Given structured output instructions, the parser interprets the format line by line with triples like `(Entity1, relation, Entity2)`. We implement regex or string splitting to extract triple components. From `(diabetes, causes, peripheral neuropathy)`", the parser strips parentheses and splits by comma, yielding subject = diabetes", relation = causes", object = peripheral neuropathy". We trim whitespace and ensure text matches the original note.

- **Entity Matching:** We cross-match extracted entity text with NER-recognized entities. Each Entity1 and Entity2 should correspond to NER-identified mentions. For minor variations (model outputs diabetes" but text had diabetes mellitus"), we match by substring or linked CUI since both share concepts. For entities entirely absent from text (likely hallucinations), we **discard that triple**. This post-hoc validation filters extraneous tokens/entities (Ji et al., 2023).

- **Relation Validation:** We validate relation parts, checking for reasonable expressions (verbs or short phrases) like causes," treats," "indicates." We flag lengthy or dubious relation phrases and ensure relations aren't empty or identical to entities. Malformed outputs like `(X, X, Y)` are removed as nonsensical.

- **Duplication and Uniqueness:** The parser handles duplicates when LLMs list relationships twice or express them similarly. We canonicalize triples (sorting/lowercasing relation phrases) and use sets for uniqueness. Most relations are directed (cause vs caused-by), so we treat `(diabetes, causes, neuropathy)` as distinct from `(neuropathy, causes, diabetes)`.

- **Confidence and Post-Filtering:** Each triple inherits confidence estimated from model reliability and keyword presence. Relations directly from text ("caused") have high confidence; inferred relations have lower confidence. MedGemma tends to stick closer to text while general models infer more, so we weight MedGemma triples slightly higher. We insert triples above basic confidence thresholds but tag questionable ones for potential analysis exclusion.

As a result of parsing and validation, we obtain a **clean set of (Subject, Relation, Object)** triples, each linked to UMLS concept IDs via the subject and object. For example, "(diabetes mellitus, causes, peripheral neuropathy)" becomes (CUI C0011849, *causes*, CUI C0031117) after we replace the text with the corresponding CUIs from the earlier linking stage. These are now ready to be ingested into the knowledge graph. The validation steps, while reducing quantity slightly, ensure that the *quality of relationships is high* – only those supported by the text and recognized entities are kept. This mitigates the risk of hallucinations or errors from the LLM contaminating the knowledge base. By designing the prompt and parser in tandem, we effectively constrain the LLM's output and then double-check it, achieving a balance between **completeness and accuracy** in relationship extraction.

3.5 Knowledge Graph Construction

3.5.1 Graph Schema Design

With a list of extracted entities (linked to UMLS) and relations, we define a schema for how these should be represented in the knowledge graph. The schema outlines what node types and relationship types exist, and what properties they carry, ensuring the graph is both expressive and normalized.

- **Node Types and Properties:** In our design, each **medical entity** becomes a node in the graph. Rather than having dozens of node categories for each fine-grained entity type, we opt for a unified node label, say `:MedicalEntity`, with properties to encode its attributes. Every node has at least:

- `cui`: the UMLS Concept Unique Identifier (as a string). This is the primary key for the node; we assume one node per unique CUI. Using CUI ensures that if the same concept appears in multiple notes, they map to the same node.
- `name`: a canonical name for the concept. We use the preferred name from UMLS for that CUI (e.g., C0011849 → "Diabetes Mellitus"). This makes the graph human-readable.
- `sem_type`: the semantic type or category of the concept, as defined in UMLS (e.g., T047 for "Disease or Syndrome", T121 for "Pharmacologic Substance"). We might also include a more readable form of the semantic type (like a label "Disease"). This property allows filtering or subtyping of nodes by broad category (e.g., query all nodes that are diseases).
- Optionally, we store other metadata if available: e.g., `definition` (a short definition from UMLS), `aliases` (synonyms), etc. We did not fully exploit these in our core pipeline due to size concerns, but including definitions could help if one were to use the graph for reasoning or verification.

We did consider splitting node labels by major semantic group (e.g. `:Disease`, `:Drug`, `:Procedure` labels). This would mirror domain ontology classes and perhaps optimize certain queries. However, UMLS semantic types are numerous, and a node can have multiple semantic types. For simplicity, we use a single label and rely on the `sem_type` property for distinguishing types. This decision keeps the schema flexible and avoids schema changes when new types appear.

- **Relationship Types:** Each extracted relation becomes an edge in the graph connecting two MedicalEntity nodes. We capitalize and possibly normalize the relation phrase to define the **relationship type**. For example, "causes" may become a `:CAUSES` relationship type in Neo4j. If a relation phrase is longer (more than one word), we either use it verbatim (spaces are allowed in Cypher if quoted) or convert to CamelCase or snake_case (e.g. "side effect of" could be stored as `:SIDE_EFFECT_OF`). We compiled a list of relation types we expect from our model outputs. Many will be symmetric inverses (e.g. "treats" vs "treated_by"). We decided to store only one direction as the relation type that was extracted. In cases where the inverse makes sense, Neo4j can query the reverse direction without duplicating edges (by traversing in reverse). For instance, if we have `(:Drug)-[:TREATS]->(:Disease)`, we don't need a separate `:TREATED_BY` edge, as the inverse can be inferred in queries. Thus, our relationship types are directed as per the text's implication. Some examples of relationship types in our schema:

  - `CAUSES` – from condition to outcome (disease to complication).
  - `TREATS` – drug or procedure to condition.
  - `ASSOCIATED_WITH` – a general link if model says "X is associated with Y".
  - `INDICATES` – symptom or test result indicating a condition.

  These are not fixed by the system initially; instead, they emerge from the model output. We then standardize them as needed. If synonyms appear (e.g. "leads to" vs "causes"), we may choose to map them to one canonical relation type for consistency.

- **Graph Orientation and Context:** Each edge could also carry a `context` or `source` property indicating from which document (or sentence) it was derived. This is valuable if we want to trace back the provenance of a relationship. In our implementation, we include a property `pmid` or `sentence_id` on the relationship to record this. For example,

  ```
  (TNF-alpha)-[CAUSES {source:‘‘PMID12345’’}]->(Inflammation)
  ```

  tells us that the relation was found in PubMed document 12345. This is useful for downstream validation or if we want to retrieve the original evidence for a given edge. We also considered temporal context (if documents have timestamps and the relation is time-bound) but our data does not deeply explore temporality, so we left that out of scope.

The resulting schema is a **property graph** schema common in biomedical KGs: a single dominant node type for concepts connected by various relation types. This design aligns with other healthcare knowledge graphs (e.g., HetioNet or OpenBioLink) which also have nodes for entities and typed edges for relationships, albeit those are often predefined by ontologies (Himmelstein et al., 2017). In our case, the schema is partly *emergent* (relation types come from text) and partly *ontological* (node identities come from UMLS). By mapping to UMLS, we ensure the graph can be merged or aligned with existing knowledge sources. If needed, one could enrich this graph by pulling in more UMLS connections (like hierarchical relations “isa” between concepts), but our focus is on the information extracted directly from biomedical literature (Cowell and Smith, 2020).

In summary, the schema is designed to capture the essential pieces: **unique medical entities** (with their standard identifiers and categories) and **meaningful relationships** between them as observed in biomedical text. It strikes a balance between specificity (not losing detail of relation phrases) and interoperability (using global IDs for concepts). This schema will support the queries and analyses described in later chapters, such as counting unique entities, listing all relations of a certain type, or measuring graph connectivity.

3.5.2 Cypher Query Generation and Storage

Once we have the schema and the list of nodes and relations (with their properties) ready, we proceed to create the knowledge graph in a Neo4j database. We use the Cypher query language for graph database operations, generating queries programmatically for each element. The process is as follows:

- **Node Merge Queries:** For each unique entity (identified by CUI) extracted from the notes, we generate a Cypher `MERGE` query to ensure a corresponding node exists in the database. We prefer `MERGE` over `CREATE` to avoid duplicate nodes, as the same entity may appear multiple times. For example, for a concept C0011849 (Diabetes Mellitus) with name “Diabetes Mellitus” and sem_type “Disease or Syndrome”, we produce:

```
MERGE (n:MedicalEntity {cui: "C0011849"})
  ON CREATE SET n.name = "Diabetes Mellitus",
                n.sem_type = "Disease or Syndrome";
```

This query checks if a node with `cui` = C0011849 exists. If not, it creates one and sets its properties. If it exists, it leaves it as is (or we could optionally update the name to ensure consistency, but typically the first creation suffices). We do this for each distinct CUI from our list of entities. These queries are often batched for efficiency (we can combine multiple MERGEs in one transaction, or send them sequentially).

- **Relationship Creation Queries:** For each relationship triple (with source CUI, relation type, target CUI), we generate a `MERGE` or `CREATE` query for the edge. We use `MERGE` similarly to avoid duplicating the same edge if processed twice. However, we also include uniqueness by context if needed. The pattern looks like:

```
MATCH (a:MedicalEntity {cui:"C0011849"}),
      (b:MedicalEntity {cui:"C0031117"})
MERGE (a)-[r:CAUSES]->(b)
  ON CREATE SET r.source = "Note42"
```

This query first finds the two nodes by their CUIs (we assume they've been created by the previous step). Then it merges a `:CAUSES` relationship from the first to the second. If the relationship did not exist, it will be created and we set a property `source` to "Note42" (for example). If it already exists (meaning we encountered the same relation earlier, perhaps from another sentence or note), the `MERGE` will match it and do nothing else. One caveat: In Neo4j, `MERGE` on a relationship without specifying all properties could merge even if *source* differs, which might or might not be desired. If we want multiple provenance, we might allow duplicates distinguished by `source`. In our case, we decide that the existence of an edge means the two concepts are related; we don't create duplicate edges for multiple notes, but we could append sources. (One could model sources as an array property or connect a `:Occurrence` node, but we keep it simple.)

- **Graph Storage:** The Neo4j database stores the resulting knowledge graph persistently. We verify the storage by running sample queries. For instance, after insertion, a query like:

```
MATCH (n:MedicalEntity)-[r]->(m:MedicalEntity)
RETURN n.name, type(r), m.name LIMIT 5;
```

would return some sample triples, confirming data presence. We also check that no unintended duplicates exist: e.g., each CUI yields exactly one node (we rely on the MERGE logic for that). The use of an **ACID transactional database** like Neo4j ensures that even if our batch insertion is interrupted, we won't end up with partial duplicates; the transactions handle consistency.

One benefit of using Neo4j and Cypher is the ability to later query complex patterns, such as *"find all drugs that treat complications of diabetes"* in a relatively straightforward graph traversal query. The choice of Cypher is natural since Neo4j is one of the most popular graph databases, and it has been used in multiple healthcare knowledge graph projects for its robust query capabilities (Neo4j, Inc., 2023). Neo4j's flexibility also means we could augment the graph with new node types or relationships easily if our schema evolves.

Finally, it's worth noting that we considered the alternative of using a RDF triple store or an RDF representation (with entities as subjects/objects and relations as predicates). We opted for Neo4j's labeled property graph model for simplicity and familiarity. Our Cypher generation approach is straightforward, but it could be automated with an Object-Graph-Mapper or by using libraries like Py2neo or Neo4j's bulk import tool if scaling up. In this thesis context, generating explicit Cypher statements gave us fine control and transparency over the insertion process, which was helpful in debugging and verifying each step of the pipeline.

By the end of this stage, the unstructured text has been fully transformed: we now have a **populated knowledge graph** where each node is a medical concept (with context from a biomedical document) and each edge encodes a relationship that was described in the text. This graph is ready to be analyzed for insights, queried for specific patterns, and evaluated against our research questions.

3.6 Implementation Optimization

3.6.1 Parallel Processing and Resource Management

Building the pipeline in a Jupyter notebook environment (on Apple Silicon hardware) required careful optimization to handle the computational load of NLP tasks and large knowledge bases. We implemented several strategies to improve runtime and manage memory:

- **Parallel Processing:** Many pipeline steps execute in parallel, especially at document level since biomedical document processing is mostly independent. We used Python multiprocessing pools or joblib to distribute work across CPU cores, spawning worker processes for NER→Linking→Relation Extraction sequences. This achieved near-linear throughput scaling until shared resource contention emerged. SciSpacy's UMLS linker posed challenges - loading the 1GB knowledge base per process is expensive. We balanced performance by parallelizing at coarse levels (processing 3 documents concurrently) without overwhelming the system (McKerns, Aivazis, and Scherer, 2021).

- **Batching of Model Inference:** GLiNER NER and LLM models can run on text batches rather than single documents. For NER, combining short documents into batches significantly increased GPU utilization. HuggingFace transformer backend supports batching - we used 8 sentences per inference call, amortizing overhead while ensuring sentences from different documents weren't mixed confusingly. For LLMs (MedGemma/Gemma), batching is trickier due to large prompt+output sizes, so we processed relations sequentially per document, compensating through process-level parallelization when memory allowed.

- **Execution Mode – In-Notebook vs. External:** Jupyter notebooks offer convenience but have limitations. We optimized by offloading long-running tasks to external scripts when needed, like using Neo4j's bulk import tool for heavy database insertion. Our moderate dataset size was manageable within notebooks through chunking and optimizations. We scheduled long steps (LLM processing) to save outputs to disk, preventing progress loss from kernel restarts during large memory tasks.

Through parallelism we improved speed and through careful memory management we averted crashes, creating a computationally intensive but research-feasible

pipeline. (GLiNER and linking are much faster, making LLM the bottleneck). By monitoring resource usage and optimizing where possible, we ensure the methodology is both theoretically sound and practically executable within our hardware constraints.

Overall, the methodology chapter has described how each component of our system works and how they come together efficiently. In the subsequent chapters, we will assess how well this methodology performs, our results by comparing the chosen models, and discuss the implications of this approach in the healthcare context. The rigorous design and optimization steps we've undertaken here lay the groundwork for those analyses, and they contribute to the **novel pipeline** we developed for turning unstructured biomedical text into an actionable knowledge graph.

Chapter 4

Results, Evaluation and Discussion

This chapter presents a comprehensive evaluation of the multi-model biomedical information extraction pipeline developed in this thesis. We evaluate each component systematically: named entity recognition using GLiNER, relationship extraction comparing Gemma and MedGemma models, and the end-to-end pipeline performance for knowledge graph construction. The evaluation is conducted on the BioRED dataset, providing gold-standard annotations for both entity recognition and relationship extraction tasks. Our findings reveal significant challenges in biomedical relationship extraction while demonstrating the potential of multi-model approaches for transforming unstructured biomedical text into structured knowledge graphs.

4.1 Experimental Setup

4.1.1 BioRED Dataset and Evaluation Metrics

Our evaluation is conducted on the **BioRED dataset** (Luo et al., 2022), a comprehensive corpus specifically designed for biomedical relation extraction evaluation. BioRED contains manually annotated PubMed abstracts covering diverse biomedical research areas with multiple entity types and relationship annotations. The dataset provides a robust foundation for evaluating our multi-model pipeline's performance across different biomedical text processing tasks.

Dataset Statistics: The BioRED dataset consists of 600 documents split across three sets: 400 training documents, 100 development documents, and 100 test documents. For our evaluation, we focus primarily on the test set to provide unbiased performance assessment. The dataset contains over 20,000 entity mentions across six entity types: GeneOrGeneProduct, DiseaseOrPhenotypicFeature, ChemicalEntity, SequenceVariant, CellLine, and OrganismTaxon. Additionally, it includes approximately 6,000 relationship annotations spanning eight relation types: Association, Positive_Correlation, Negative_Correlation, Bind, Comparison, Cotreatment, Drug_Interaction, and Conversion.

Evaluation Metrics: We employ standard information extraction evaluation metrics to assess pipeline performance. For named entity recognition, we calculate **precision**, **recall**, and **F1 score** using three matching strategies: exact matching (requiring precise boundary alignment), partial matching (allowing overlapping boundaries), and text-based matching (focusing on entity surface forms). For relationship extraction, we use precision, recall, and F1 score at the relation level, where a prediction is considered correct only if both entities and the relation type match the gold standard.

The choice of BioRED enables direct comparison with existing biomedical NLP systems and provides realistic evaluation conditions. The dataset's annotation quality and comprehensive coverage of biomedical relationships make it particularly suitable for evaluating knowledge graph construction pipelines. Our evaluation framework systematically measures each pipeline component's contribution to overall system performance, enabling identification of bottlenecks and optimization opportunities.

4.1.2 Model Configurations and Prompt Strategies

Our experimental setup evaluates multiple model configurations to understand the impact of domain specialization and prompt engineering on biomedical information extraction performance. We configure the pipeline to test different approaches systematically while maintaining consistent preprocessing and evaluation procedures.

Named Entity Recognition Configuration: We employ the GLiNER-BioMed model (Ihor/gliner-biomed-bi-large-v1.0) for named entity recognition, configured with three different confidence thresholds to analyze precision-recall trade-offs. The **default threshold** (0.5) represents standard model operation, the **low threshold** (0.3) maximizes recall by accepting more tentative predictions, and the **high threshold** (0.7) prioritizes precision by filtering uncertain predictions. This threshold analysis reveals the model's calibration and optimal operating points for different application scenarios.

Relationship Extraction Model Variants: We evaluate six different configurations combining two base models with three prompt strategies. The base models are **Gemma-3-4b-it** (general-purpose instruction-tuned model) and **MedGemma-3-4b** (medical domain-specialized variant). For each model, we test three prompt strategies:

- **Basic Prompting:** Simple task description requesting relationship extraction without examples or structured formatting guidance.
- **Few-shot Prompting:** Includes 2-3 demonstration examples showing desired input-output format and relationship types.
- **Structured Prompting:** Detailed instructions with explicit output formatting requirements, entity type constraints, and relationship category guidelines.

This configuration matrix allows systematic comparison of domain specialization effects (Gemma vs MedGemma) and prompt engineering impact (basic vs few-shot vs structured). The experimental design isolates these factors while maintaining consistent evaluation conditions, enabling reliable performance attribution to specific design choices.

Evaluation Protocol: Each model configuration processes the same BioRED test set under identical conditions. We record computational requirements (inference time, memory usage) alongside accuracy metrics to assess practical deployment feasibility. The evaluation framework captures both component-level performance (NER accuracy, relation extraction F1) and end-to-end pipeline effectiveness (knowledge graph completeness, entity linking success rates). This comprehensive evaluation approach provides insights into real-world system behavior beyond isolated component performance.

4.2 Named Entity Recognition Performance

4.2.1 GLiNER Results: Threshold Impact and Matching Strategies

The GLiNER-BioMed model demonstrates varying performance characteristics across different confidence thresholds and matching criteria, revealing important trade-offs between precision and recall in biomedical named entity recognition. Our systematic evaluation across three threshold settings provides insights into optimal model configuration for different application requirements.

Default Threshold Performance (0.5): At the default threshold, GLiNER-BioMed achieves moderate precision with limited recall across all matching strategies. With **exact matching**, the model attains 62.36% precision but only 19.92% recall, resulting in an F1 score of 30.19%. This indicates that while the model's predictions are reasonably accurate, it misses a significant portion of true entities. The **partial matching** strategy shows improvement with 69.35% precision and 22.15% recall (F1: 33.58%), suggesting that many model predictions have correct entity boundaries but differ slightly from gold standard annotations. Interestingly, **text-based matching** shows reduced performance (54.52% precision, 19.73% recall, F1: 28.97%), indicating that surface form matching introduces additional complexity.

Low Threshold Impact (0.3): Reducing the confidence threshold to 0.3 significantly improves recall at the cost of precision, as expected. Exact matching achieves 55.53% precision and 31.26% recall (F1: 40.00%), representing a substantial recall improvement of over 10 percentage points. Partial matching reaches 62.01% precision with 34.91% recall (F1: 44.67%), the highest F1 score achieved in our evaluation. This suggests that lowering the threshold captures more true entities while maintaining reasonable prediction quality. The text-based matching continues to show challenges with 47.04% precision and 32.56% recall (F1: 38.48%).

High Threshold Conservative Approach (0.7): The high threshold configuration prioritizes precision over recall, achieving 69.91% precision but only 8.54% recall for exact matching (F1: 15.23%). While this configuration produces highly confident predictions, it misses the majority of entities, making it unsuitable for comprehensive knowledge extraction. Partial matching slightly improves recall to 9.22% while maintaining 75.46% precision (F1: 16.44%), but the overall performance remains limited for practical applications.

These results demonstrate that GLiNER-BioMed exhibits **conservative behavior** in biomedical entity recognition, preferring high-confidence predictions over comprehensive coverage. The optimal threshold depends on application requirements: clinical decision support systems might prefer high precision (threshold 0.7), while knowledge discovery applications benefit from balanced performance (threshold 0.3-0.4). The consistent performance gap between partial and exact matching suggests that boundary detection remains challenging, possibly due to variations in annotation standards or entity mention complexity.

Comprehensive Performance Summary: Table 4.1 presents the complete NER evaluation results across all model configurations and matching strategies, ordered by F1 score performance. The results clearly demonstrate the performance hierarchy among different threshold settings and matching approaches.

The table confirms that the **gliner_biomed_low_threshold** configuration with partial matching achieves the best overall performance (F1: 0.447), representing the optimal balance between precision (0.620) and recall (0.349) for comprehensive biomedical entity extraction. This configuration should be recommended for applications prioritizing entity coverage over ultra-high precision.

TABLE 4.1: NER Evaluation Summary - GLiNER-BioMed Performance Across Configurations

Model	Matching	F1	Precision	Recall
gliner_biomed_low_threshold	partial	0.447	0.620	0.349
gliner_biomed_low_threshold	exact	0.400	0.555	0.313
gliner_biomed_low_threshold	text	0.385	0.470	0.326
gliner_biomed_default	partial	0.336	0.694	0.221
gliner_biomed_default	exact	0.302	0.624	0.199
gliner_biomed_default	text	0.290	0.545	0.197
gliner_biomed_high_threshold	partial	0.164	0.755	0.092
gliner_biomed_high_threshold	exact	0.152	0.699	0.085
gliner_biomed_high_threshold	text	0.146	0.654	0.082

4.2.2 Error Analysis and Entity Boundary Challenges

Detailed error analysis reveals systematic patterns in GLiNER-BioMed's prediction failures, providing insights into biomedical NER challenges and potential improvement directions. The analysis examines false positives, false negatives, and boundary detection issues across different entity types and text contexts.

Boundary Detection Issues: The performance difference between partial and exact matching (approximately 3-4 F1 points) indicates systematic boundary detection challenges. Common patterns include: **abbreviated forms** where the model predicts "TNF" but the gold standard annotates "TNF-α", **compound terms** where predictions capture partial phrases (e.g., "heart failure" vs "congestive heart failure"), and **modifier inclusion** where clinical context affects boundary determination (e.g., "severe diabetes" vs "diabetes"). These boundary issues particularly affect ChemicalEntity and DiseaseOrPhenotypicFeature categories, which often involve complex terminological variations.

Entity Type Distribution: Different entity types show varying recognition success rates. GeneOrGeneProduct entities typically achieve higher precision due to standardized naming conventions (e.g., "BRCA1", "p53"), while DiseaseOrPhenotypicFeature entities show more variability due to descriptive terminology and synonym usage. ChemicalEntity recognition faces challenges with drug names, chemical compounds, and dosage expressions that may include numerical values and units. SequenceVariant entities present particular difficulties due to specialized notation (e.g., "V244M", "c.1234G>A") that requires domain-specific understanding.

Context Sensitivity: Error analysis reveals that GLiNER-BioMed struggles with context-dependent entity recognition. Ambiguous terms like "cold" (temperature vs. common cold), "culture" (cell culture vs. bacterial culture), or "positive" (test result vs. correlation direction) require surrounding context for correct classification. The model shows limited ability to leverage sentence-level context for disambiguation, suggesting potential improvements through context-aware training or post-processing rules.

Recall Limitations: The consistently low recall (20-35%) across threshold settings indicates that GLiNER-BioMed misses many true entities, particularly less common terms, newly coined expressions, and entities expressed through paraphrasing rather than standard terminology. This limitation significantly impacts downstream relationship extraction and knowledge graph completeness, as missing entities cannot participate in extracted relationships. The recall limitation represents a critical

bottleneck for comprehensive biomedical knowledge extraction applications.

These error patterns highlight the complexity of biomedical entity recognition and the need for domain-specific improvements. While GLiNER-BioMed demonstrates reasonable precision for confident predictions, the recall limitations and boundary detection challenges suggest that combining multiple NER approaches or implementing post-processing refinements could improve overall pipeline performance.

4.3 Relationship Extraction Performance

4.3.1 Gemma vs MedGemma: Strategy Comparison and Low F1 Analysis

The relationship extraction evaluation reveals significant performance challenges across all model configurations, with F1 scores consistently below 10% indicating fundamental difficulties in biomedical relationship extraction from unstructured text. Despite these low absolute scores, meaningful differences emerge between general and domain-specific models, providing insights into the value of medical specialization.

Model Comparison Overview: Across all prompt strategies, standard Gemma models consistently outperform their MedGemma counterparts, contradicting initial expectations that domain specialization would improve biomedical relationship extraction. The **Gemma few-shot** configuration achieves the highest performance with 8.18% precision, 8.20% recall, and 8.19% F1 score, compared to the best MedGemma configuration (few-shot) achieving 8.70% precision, 5.90% recall, and 7.03% F1 score. This pattern holds across all three prompt strategies, suggesting that general language understanding may be more valuable than domain-specific training for this particular relationship extraction task.

Precision vs Recall Trade-offs: The results reveal distinct behavioral patterns between model types. MedGemma variants tend to achieve higher precision but significantly lower recall, indicating more conservative prediction behavior. For example, MedGemma few-shot achieves 8.70% precision compared to Gemma few-shot's 8.18%, but MedGemma's recall drops to 5.90% versus Gemma's 8.20%. This suggests that medical domain training may lead to overly cautious relationship prediction, missing many true relationships while maintaining reasonable prediction accuracy for identified relationships.

Prompt Strategy Effects: Few-shot prompting consistently produces the best results across both model families, indicating that demonstration examples effectively guide model behavior for relationship extraction. Basic prompting achieves moderate performance, suggesting that even simple task descriptions enable reasonable relationship extraction. Structured prompting unexpectedly produces the lowest performance across both model families, potentially due to overly rigid constraints that limit the models' natural language understanding capabilities.

Performance Analysis: The consistently low F1 scores (5-8%) across all configurations indicate that biomedical relationship extraction remains a challenging task even for large language models. Several factors contribute to these difficulties: **annotation complexity** where gold-standard relationships require deep biomedical understanding, **implicit relationships** that are not explicitly stated in text, **entity linking dependencies** where incorrect entity recognition propagates to relationship extraction errors, and **relationship type ambiguity** where similar semantic relationships map to different gold standard categories.

The superior performance of general Gemma models suggests that broader language understanding and reasoning capabilities may be more important than domain-specific medical knowledge for relationship extraction tasks. This finding challenges assumptions about domain specialization benefits and suggests that scaling general capabilities might be more effective than narrow domain tuning for complex biomedical NLP tasks.

Comprehensive Performance Comparison: Table 4.2 presents the complete relationship extraction evaluation results across all model configurations and prompt strategies, ordered by F1 score performance. The results demonstrate the clear performance hierarchy among different model-prompt combinations.

TABLE 4.2: Relationship Extraction Evaluation Summary - Gemma vs MedGemma Performance

Model	F1	Precision	Recall	Docs
gemma_few_shot	0.082	0.082	0.082	100
gemma_basic	0.073	0.073	0.073	100
medgemma_few_shot	0.070	0.087	0.059	100
gemma_structured	0.069	0.075	0.065	100
medgemma_basic	0.063	0.084	0.050	100
medgemma_structured	0.053	0.068	0.043	100

The evaluation confirms that **gemma_few_shot** achieves the best overall performance (F1: 0.082) among all configurations tested. However, all models fall significantly below the minimum viable F1 score threshold of 0.40, highlighting the fundamental challenges in biomedical relationship extraction. The consistent pattern of Gemma models outperforming MedGemma variants across all prompt strategies reinforces the finding that general language capabilities may be more valuable than domain-specific medical training for this complex task.

4.3.2 Model Limitations and Output Quality Issues

Detailed analysis of model outputs reveals systematic limitations that explain the low F1 scores and highlight areas requiring improvement for practical biomedical relationship extraction applications. These limitations span output formatting, relationship identification accuracy, and adherence to task constraints.

Output Formatting Challenges: Both Gemma and MedGemma models struggle with consistent output formatting despite explicit instructions. Common issues include **malformed triples** where models produce incomplete or incorrectly structured relationships (e.g., missing entities or relation types), **entity naming inconsistencies** where models use variations of entity names not matching input text exactly, and **extraneous content** where models include explanatory text or reasoning alongside requested triple outputs. These formatting issues require substantial post-processing and reduce the reliability of automated knowledge extraction.

Hallucination and Inference Issues: Large language models demonstrate tendency to infer relationships not explicitly stated in the text, creating both opportunities and challenges for biomedical knowledge extraction. While some inferences reflect valid biomedical knowledge (e.g., inferring that a drug treats a condition when the text mentions prescription context), others introduce errors through incorrect assumptions or over-generalization. MedGemma models show slightly higher rates

of medically plausible but textually unsupported inferences, suggesting that domain knowledge can lead to hallucination problems in structured extraction tasks.

Relationship Type Granularity: The BioRED dataset defines specific relationship types (Association, Positive_Correlation, Negative_Correlation, etc.) that require precise semantic distinction. Models frequently confuse similar relationship types, particularly between "Association" and "Positive_Correlation" or between "Cotreatment" and "Drug_Interaction." This granularity challenge reflects the difficulty of mapping natural language expressions to formal relationship taxonomies, a key challenge for automated knowledge base construction.

Entity Alignment Problems: Relationship extraction accuracy depends critically on correct entity identification and alignment. Models often predict relationships using entity phrasings that differ from NER outputs, creating alignment challenges during knowledge graph construction. For example, if NER identifies "diabetes mellitus" but the relationship extractor outputs "diabetes," post-processing must resolve this discrepancy or risk losing the relationship. This entity alignment problem compounds errors from both NER and relationship extraction stages.

Repetition and Redundancy: Models occasionally produce duplicate or near-duplicate relationships, particularly when processing longer texts with multiple entity mentions. This redundancy creates noise in extracted knowledge and requires deduplication procedures that may inadvertently remove valid relationships between recurring entities. The repetition issue appears more pronounced in basic prompting conditions, suggesting that structured guidance helps maintain output quality.

Model Size Constraints: A significant limitation of our evaluation is the use of relatively small 4B parameter models (Gemma and MedGemma), which may contribute to the low F1 scores observed. The original BioRED paper reports much higher performance using larger models and specialized architectures, with state-of-the-art systems achieving F1 scores above 40% compared to our 5-8%. While 4B parameter models offer practical advantages for deployment (lower memory requirements, faster inference), they may lack the capacity to capture the complex patterns necessary for accurate biomedical relationship extraction. Larger models (70B+ parameters) or models specifically fine-tuned on biomedical relation extraction tasks would likely achieve substantially better performance, suggesting that model scale remains a critical factor for this challenging task.

These issues highlight the gap between large language model capabilities and the precision requirements of structured knowledge extraction. While models demonstrate impressive language understanding, the systematic formatting and accuracy challenges indicate that relationship extraction for knowledge graph construction requires specialized approaches beyond standard language modeling techniques. The findings suggest that hybrid approaches combining language models with rule-based post-processing or specialized training objectives and using larger relationship extraction models might be necessary for practical biomedical relationship extraction applications.

4.4 End-to-End Pipeline Evaluation

4.4.1 Processing Efficiency and Scalability

The pipeline's computational performance and scalability characteristics determine practical deployment feasibility for large-scale biomedical text processing applications. Our evaluation assesses processing times, resource requirements, and scalability bottlenecks across pipeline components.

Component-Level Performance: Processing times vary significantly across pipeline stages, with relationship extraction representing the primary computational bottleneck. **GLiNER NER** processes documents efficiently at approximately 2-5 seconds per abstract on Apple Silicon hardware, depending on text length and entity density. **SciSpacy entity linking** adds minimal overhead (0.5-1 second per document) due to efficient TF-IDF similarity computation and optimized UMLS index structures. **Large language model inference** dominates processing time, requiring 15-45 seconds per document depending on model size, prompt complexity, and generated output length.

Memory Requirements: The pipeline's memory footprint reflects the size of loaded models and knowledge bases. GLiNER requires approximately 2-3 GB of GPU memory when loaded, while SciSpacy's UMLS linker consumes 1-2 GB of RAM for knowledge base indexing. Large language models (Gemma/MedGemma 4B parameters) require 8-12 GB of GPU memory for efficient inference, representing the largest memory demand. Total system memory requirements range from 12-18 GB for optimal performance, limiting deployment to well-equipped hardware configurations.

Scalability Analysis: Linear scaling analysis reveals that processing 100 BioRED documents requires approximately 160-180 minutes depending on model configuration and hardware specifications. This suggests that large-scale processing (e.g., entire PubMed subsets) would require distributed computing approaches or specialized optimization strategies. The relationship extraction stage's computational intensity creates a significant scalability bottleneck that would need addressing for production deployment.

Optimization Opportunities: Several optimization strategies could improve pipeline efficiency: **model quantization** to reduce memory requirements and inference time, **batch processing** to amortize model loading overhead across multiple documents, **distributed processing** to parallelize document-level computation, and **selective processing** to apply expensive models only to high-value content sections. The modular pipeline architecture facilitates such optimizations without requiring fundamental redesign.

Deployment Considerations: Practical deployment requires balancing accuracy requirements with computational constraints. High-throughput applications might benefit from faster but less accurate models or simplified relationship extraction approaches. Research applications requiring comprehensive extraction might justify longer processing times for improved coverage and accuracy. The pipeline's flexibility allows configuration optimization based on specific deployment requirements and available computational resources.

4.5 Discussion

4.5.1 Key Findings and Clinical Applicability

The comprehensive evaluation of our multi-model biomedical information extraction pipeline reveals both promising capabilities and significant challenges for transforming unstructured biomedical text into actionable knowledge graphs. The findings provide important insights into current limitations and future directions for biomedical NLP applications.

Performance Summary and Implications: The evaluation demonstrates that while individual pipeline components show reasonable performance for model size in isolation, the end-to-end system faces substantial challenges in comprehensive knowledge extraction. GLiNER-BioMed achieves moderate precision (55-70%) but limited recall (20-35%) for named entity recognition, while relationship extraction remains particularly challenging with F1 scores below 10% across all model configurations. These results indicate that current approaches, while technically feasible, require significant model improvement before deployment in critical clinical applications.

Domain Specialization Insights: The unexpected finding that general Gemma models outperform domain-specific MedGemma variants challenges assumptions about the value of medical specialization for relationship extraction tasks. This suggests that broad language understanding and reasoning capabilities may be more important than domain-specific training for complex biomedical relationship identification. However, the overall low performance across both model types indicates that relationship extraction remains fundamentally challenging regardless of domain specialization approach.

Clinical Applicability Assessment: The current pipeline performance limits immediate clinical applicability, particularly for applications requiring comprehensive knowledge extraction or high recall. However, specific use cases might benefit from the high-precision, low-recall characteristics observed in our evaluation. **Literature screening** applications could leverage the pipeline's ability to identify high-confidence relationships for initial filtering, reducing manual review requirements. **Knowledge base enrichment** could benefit from the pipeline's entity linking capabilities and standardized concept identification, even with incomplete relationship extraction. **Hypothesis generation** might utilize extracted relationships as starting points for deeper investigation, accepting lower recall in exchange for automated processing capabilities.

Technical Architecture Strengths: Despite performance limitations, the multi-model pipeline architecture demonstrates several advantages. The **modular design** enables independent optimization of each component and facilitates integration of improved models as they become available. The **UMLS integration** provides interoperability with existing medical knowledge bases and standards. The **flexible configuration** options allow adaptation to different accuracy-efficiency trade-offs based on application requirements. These architectural strengths suggest that the framework provides a solid foundation for future improvements.

Chapter 5

Conclusions and Future Work

This thesis presented a comprehensive multi-model approach for transforming unstructured biomedical text into actionable knowledge graphs through the integration of named entity recognition, entity linking, relationship extraction, and graph construction techniques. This final chapter summarizes the key contributions of this work and outlines promising directions for future research.

5.1 Summary of Contributions

5.1.1 Technical Contributions

Multi-Model Pipeline Architecture: Developed an end-to-end pipeline integrating GLiNER for named entity recognition, SciSpacy's UMLS linking for entity normalization, and large language models (Gemma/MedGemma) for relationship extraction with modular architecture for flexible component substitution.

Prompt Engineering for Relationship Extraction: Developed and evaluated multiple prompting strategies for biomedical relationship extraction, comparing Gemma vs. MedGemma configurations to identify effective approaches for semantic relationship extraction.

Scalable Graph Construction: Implemented efficient Cypher query generation and batch processing for Neo4j knowledge graphs with parallel processing and resource management optimizations for large-scale biomedical document processing.

Comprehensive Evaluation Framework: Established evaluation methodology using BioRED dataset with multiple matching strategies (exact, partial, text-based) and metrics for pipeline component assessment.

5.1.2 Theoretical Contributions

Model Specialization Analysis: Comparison between Gemma and MedGemma revealed that domain specialization does not automatically improve performance in complex relationship extraction tasks, with implications for specialized biomedical language model development.

Knowledge Graph Integration Paradigm: Demonstrated effective structuring and querying of extracted biomedical knowledge within graph databases, providing a foundation for biomedical knowledge discovery and reasoning systems.

5.2 Future Research Directions

Based on the findings and limitations identified in this thesis, several promising avenues for future research emerge:

5.2.1 Model Scale Improvements

The most significant opportunity for performance improvement lies in scaling up the models throughout the pipeline. Our evaluation demonstrated limited relationship extraction performance with Gemma3-4B and MedGemma3-4B models, suggesting that deploying larger language models (70B+ parameters) would likely yield substantial improvements in F1 scores due to their superior reasoning capabilities and biomedical knowledge understanding.

5.2.2 Expanding Beyond UMLS to Other Medical Knowledge Bases

Future work should explore integrating multiple biomedical ontologies simultaneously, including Gene Ontology (GO), Chemical Entities of Biological Interest (ChEBI), and Human Phenotype Ontology (HPO), to provide richer semantic annotations and enable cross-ontology relationship discovery that could reveal novel connections between different biomedical entity types (Himmelstein et al., 2017).

5.3 Final Remarks

While we successfully developed an end-to-end system integrating state-of-the-art NLP techniques, significant obstacles remain in achieving high-quality biomedical information extraction, particularly in relationship extraction performance. The modular architecture provides a platform for incorporating future advances as the field rapidly evolves.

Appendix A

Comprehensive Evaluation Results and Statistical Analysis

This appendix provides comprehensive evaluation results from our biomedical text extraction pipeline. We present detailed performance metrics for both the Named Entity Recognition (NER) component using GLiNER and the relationship extraction component using Gemma and MedGemma models. The evaluation scripts and data processing code used to generate these results are available in our GitHub repository Abioye, 2025.

A.1 BioRED Dataset Evaluation Overview

All evaluations were performed on the BioRED dataset test set consisting of 100 documents with the following characteristics:

- Total entities: 3,535
- Total relationships: 1,163
- Entity types: Gene, Disease, Chemical, Species, Cell Type, Gene Variant
- Relation types: 8 biomedical relationship types

A.2 Named Entity Recognition (NER) Performance

A.2.1 GLiNER Threshold Sensitivity Analysis

Table A.1 presents the performance metrics for GLiNER at different confidence thresholds using exact matching criteria.

TABLE A.1: GLiNER Performance at Different Confidence Thresholds (Exact Matching)

Threshold	Precision	Recall	F1 Score	TP	FP	FN	Predictions
0.3 (Low)	0.556	0.313	0.400	1105	885	2430	1990
0.5 (Default)	0.624	0.199	0.302	704	425	2831	1129
0.7 (High)	0.699	0.085	0.152	302	130	3233	432

Key observations:

- Lower thresholds improve recall but reduce precision

- The default threshold (0.5) provides a balanced trade-off
- High threshold (0.7) achieves best precision but severely limits recall

A.2.2 Matching Strategy Performance Comparison

Table A.2 compares different matching strategies for entity recognition evaluation.

TABLE A.2: GLiNER Performance Across Different Matching Strategies (Threshold=0.5)

Matching Strategy	Precision	Recall	F1 Score	TP	FP	FN
Exact	0.624	0.199	0.302	704	425	2831
Partial	0.751	0.240	0.364	848	281	2687
Text-based	0.804	0.257	0.389	908	221	2627

A.2.3 Entity Type-Specific Performance

The following section would include detailed breakdowns by entity type, however, this information requires additional analysis of the evaluation data per entity type.

A.3 Relationship Extraction Performance

A.3.1 Model Configuration Comparison

Table A.3 presents the comprehensive evaluation results for all relationship extraction model configurations.

TABLE A.3: Relationship Extraction Performance Across Model Configurations

Model	Strategy	Precision	Recall	F1	TP	FP	FN
Gemma	Basic	0.073	0.073	0.073	118	1506	1492
Gemma	Few-shot	0.082	0.082	0.082	132	1482	1478
Gemma	Structured	0.075	0.065	0.070	104	1280	1506
MedGemma	Basic	0.084	0.050	0.063	81	888	1529
MedGemma	Few-shot	0.087	0.059	0.070	95	997	1515
MedGemma	Structured	0.068	0.043	0.053	69	944	1541

Key findings:

- All models show extremely low performance (F1 < 0.1)
- Few-shot prompting slightly improves performance
- MedGemma unexpectedly underperforms compared to base Gemma
- Over 90% of relationships are missed by all configurations

A.3.2 Detailed Error Analysis

The relationship extraction component faces several critical challenges:

Output Format Issues

- Inconsistent JSON formatting despite structured prompting
- Entity hallucination - models generate entities not present in input
- Incorrect relationship type assignment
- Failure to follow prompt instructions

Model-Specific Observations

- **Gemma models**: Tend to over-generate relationships (higher FP)
- **MedGemma models**: More conservative but miss more relationships
- Both model families struggle with complex biomedical terminology

A.4 Prompting Strategy Analysis

A.4.1 Basic Prompting

```
Example output issues:
- Missing required JSON fields
- Entity names not matching input text
- Invented relationships between unrelated entities
```

A.4.2 Few-Shot Prompting

```
Improvements observed:
- Better adherence to output format
- Slight increase in recall
- Still significant entity hallucination
```

A.4.3 Structured JSON Prompting

```
Mixed results:
- Most consistent output format
- Lower overall performance
- Models struggle with schema complexity
```

A.5 Statistical Significance Tests

Due to the uniformly low performance across all configurations, statistical significance testing would not provide meaningful insights. All models perform near random baseline levels.

A.6 Performance by Relation Type

While detailed per-relation type analysis requires additional data processing, preliminary observations indicate:

- Association relationships are most frequently predicted

- Complex relationships (e.g., Cotreatment, Bind) are rarely identified
- Models show bias towards common relationship types in training data

A.7 Computational Performance Metrics

A.7.1 Processing Time Analysis

TABLE A.4: Average Processing Time per Document

Component	Average Time (s)	Documents/Hour
GLiNER NER	0.8	4500
UMLS Entity Linking	1.2	3000
Relationship Extraction	3.5	1029
Neo4j Storage	0.3	12000
Total Pipeline	**5.8**	**620**

A.8 Conclusions from Evaluation

The evaluation results highlight several critical areas for improvement:

1. **NER Performance**: GLiNER shows reasonable performance with appropriate threshold tuning
2. **Relationship Extraction**: Current approach with small language models is inadequate
3. **Model Selection**: Domain-specific models (MedGemma) do not guarantee better performance
4. **Prompt Engineering**: More sophisticated prompting strategies needed for complex tasks

A.9 Recommendations for Future Work

Based on these comprehensive evaluation results:

- Consider larger language models (GPT-4, Claude) for relationship extraction
- Implement ensemble approaches combining multiple NER models
- Develop domain-specific fine-tuning for relationship extraction
- Explore hybrid approaches combining rule-based and ML methods

Appendix B

Technical Implementation and Reproducibility Guide

This appendix provides comprehensive technical details for reproducing the biomedical text extraction pipeline, including complete prompt templates, Neo4j schema definitions, and model configurations.

B.1 System Requirements and Dependencies

B.1.1 Hardware Requirements

- GPU: NVIDIA GPU with 8GB+ VRAM (recommended) or Apple Silicon (M1/M2)
- RAM: 16GB minimum, 32GB recommended
- Storage: 50GB for models and data

B.1.2 Software Dependencies

```
Python 3.11+
transformers==4.51.3
datasets==3.6.0
spacy==3.6.1
scispacy==0.5.5
gliner==0.2.0
mlx-lm==0.15.0 (for Apple Silicon)
neo4j==5.20.0
pandas==2.0.3
numpy==1.26.4
```

B.2 GLiNER Biomedical Model Configuration

B.2.1 Model Initialization

```
from gliner import GLiNER

# Load pre-trained biomedical GLiNER model
model = GLiNER.from_pretrained("Ihor/gliner-biomed-bi-large-v1.0")

# Entity type labels for biomedical NER
labels = [
```

```
    "GeneOrGeneProduct",
    "DiseaseOrPhenotypicFeature",
    "ChemicalEntity",
    "SequenceVariant",
    "CellLine",
    "OrganismTaxon"
]
```

B.2.2 Threshold Configuration

The evaluation tested three confidence threshold settings:

- **Low threshold (0.3)**: Maximum recall, lower precision
- **Default threshold (0.5)**: Balanced performance
- **High threshold (0.7)**: Maximum precision, lower recall

```
# Entity extraction with configurable threshold
predictions = model.predict_entities(
    text,
    labels=labels,
    threshold=0.5  # Adjustable: 0.3, 0.5, or 0.7
)
```

B.2.3 Optimization Parameters

```
# Batch processing for efficiency
batch_size = 8  # Adjust based on GPU memory

# Maximum sequence length
max_length = 512  # GLiNER token limit

# Parallel processing
max_workers = 4  # For multi-threaded execution
```

B.3 Relationship Extraction Prompt Templates

B.3.1 Basic Prompting Strategy

```
def create_basic_prompt(text, entities):
    prompt = f"""Your goal is to perform a Closed Information
Extraction task on the following clinical note:

{text}

You are provided with a list of medical entities extracted
from the note:
{entities}

RELATION TYPES:
- Association: General association between entities
```

```
- Positive_Correlation: Entity1 increases/enhances Entity2
- Negative_Correlation: Entity1 decreases/inhibits Entity2
- Bind: Physical binding between entities
- Cotreatment: Entities used together in treatment
- Comparison: Comparing effectiveness of entities
- Drug_Interaction: Interaction between drugs
- Conversion: One entity converts to another

Your task is to generate high quality triplets of the form
(entity1, relation, entity2) where:
- The relationship is explicitly stated or strongly implied
- The entities are from the provided list (exact text)
- The triplets should be clinically meaningful

Please return the triplets in the following JSON format:
[
  {
    "entity1": "entity name",
    "entity2": "entity name",
    "relation": "relation_type"
  }
]"""
    return prompt
```

B.3.2 Few-Shot Prompting Strategy

```
def create_few_shot_prompt(text, entities):
    prompt = f"""Your goal is to perform a Closed Information
Extraction task on the following clinical note:

{text}

You are provided with a list of medical entities extracted
from the note:
{entities}

RELATION TYPES:
[Same as basic strategy...]

Some few-shot examples to guide you:
Example 1:
Text: "Aspirin treatment reduced inflammation in patients
       with arthritis."
Entities: aspirin (ChemicalEntity), inflammation
          (DiseaseOrPhenotypicFeature), arthritis
          (DiseaseOrPhenotypicFeature)
Relations: [{"entity1": "aspirin", "entity2": "inflammation",
            "relation": "Negative_Correlation"}]

Example 2:
Text: "The protein binds to DNA and regulates gene expression."
```

```
Entities: protein (GeneOrGeneProduct), DNA (GeneOrGeneProduct),
          gene expression (GeneOrGeneProduct)
Relations: [{"entity1": "protein", "entity2": "DNA",
            "relation": "Bind"}]

Please return the triplets in the following JSON format:
[...]"""
    return prompt
```

B.3.3 Structured JSON Prompting Strategy

```
def create_structured_prompt(text, entities):
    prompt = f"""BIOMEDICAL RELATION EXTRACTION TASK

INPUT TEXT:
{text}

AVAILABLE ENTITIES:
{entities}

RELATION TYPES:
[Same as basic strategy...]

INSTRUCTIONS:
1. Identify pairs of entities that have relationships
2. Determine the most appropriate relation type
3. Only extract relations explicitly stated or strongly implied

OUTPUT FORMAT (JSON):
[
  {
    "entity1": "exact entity text",
    "entity2": "exact entity text",
    "relation": "relation_type"
  }
]"""
    return prompt
```

B.4 Neo4j Graph Schema and Cypher Templates

B.4.1 Graph Schema Design

```
// Node Schema
(:MedicalEntity {
    name: String,           // Entity text
    cui: String,           // UMLS Concept Unique Identifier
    canonical_name: String, // UMLS canonical name
    semantic_types: String, // Pipe-separated semantic types
    linking_score: Float,   // SciSpacy confidence score
    description: String     // Entity definition
})
```

```
// Relationship Schema
()-[:RELATIONSHIP {
    type: String,          // Relationship type
    source_document: String, // Document ID
    confidence: Float      // Optional confidence score
}]->()
```

B.4.2 Entity Creation Cypher Template

```
// Create or merge entities with UMLS metadata
MERGE (e:MedicalEntity {name: $entity_name})
ON CREATE SET
    e.cui = $cui,
    e.canonical_name = $canonical_name,
    e.semantic_types = $semantic_types,
    e.linking_score = $linking_score,
    e.description = $description
ON MATCH SET
    e.cui = COALESCE(e.cui, $cui),
    e.canonical_name = COALESCE(e.canonical_name, $canonical_name)
RETURN e
```

B.4.3 Relationship Creation Cypher Template

```
// Create relationships between entities
MATCH (e1:MedicalEntity {name: $entity1_name})
MATCH (e2:MedicalEntity {name: $entity2_name})
MERGE (e1)-[r:RELATIONSHIP {type: $relation_type}]->(e2)
ON CREATE SET
    r.source_document = $document_id,
    r.timestamp = timestamp()
RETURN e1, r, e2
```

B.4.4 Query Templates for Knowledge Retrieval

```
// Find all relationships for a specific entity
MATCH (e:MedicalEntity {name: $entity_name})-[r:RELATIONSHIP]-()
RETURN e, r

// Find entities connected by specific relationship type
MATCH (e1:MedicalEntity)-[r:RELATIONSHIP {type: $rel_type}]->(e2)
RETURN e1.name, r.type, e2.name

// Find multi-hop relationships (e.g., drug interactions)
MATCH path = (e1:MedicalEntity)-[:RELATIONSHIP*1..2]-(e2:MedicalEntity)
WHERE e1.name = $start_entity AND e2.name = $end_entity
RETURN path
```

B.5 SciSpacy Entity Linking Configuration

B.5.1 Pipeline Setup

```
import spacy
import scispacy
from scispacy.linking import EntityLinker
from scispacy.abbreviation import AbbreviationDetector

# Load SciSpacy biomedical model
nlp = spacy.load("en_core_sci_lg")

# Add abbreviation detector
nlp.add_pipe("abbreviation_detector")

# Add UMLS entity linker with configuration
nlp.add_pipe("scispacy_linker", config={
    "resolve_abbreviations": True,
    "linker_name": "umls",
    "max_entities_per_mention": 3,  # Top 3 candidates
    "threshold": 0.7  # Minimum similarity score
})
```

B.5.2 Entity Linking Process

```
def link_entities(text, gliner_entities):
    """Link GLiNER entities to UMLS concepts."""
    doc = nlp(text)
    entity_links = {}

    for ent in doc.ents:
        if ent._.kb_ents:
            candidates = []
            for umls_ent in ent._.kb_ents[:3]:
                cui, score = umls_ent
                linker = nlp.get_pipe("scispacy_linker")
                kb_entity = linker.kb.cui_to_entity[cui]
                candidates.append({
                    'cui': cui,
                    'score': score,
                    'name': kb_entity.canonical_name,
                    'definition': kb_entity.definition,
                    'types': list(kb_entity.types)
                })
            entity_links[ent.text.lower()] = candidates

    return entity_links
```

B.6 Model Training and Deployment

B.6.1 Environment Setup

```
# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# or
venv\Scripts\activate  # Windows

# Install dependencies
pip install -r requirements.txt

# Download models
python -m spacy download en_core_sci_lg
```

B.6.2 Running the Pipeline For GLiNER Evaluation

```
python gliner_evaluation.py
```

B.6.3 Running the Pipeline For Gemma Evaluation

```
python gemma_evaluation.py
```

B.7 Performance Optimization

B.7.1 Caching Strategy

```
from functools import lru_cache

@lru_cache(maxsize=10000)
def cached_entity_lookup(entity_text: str) -> Dict:
    """Cache UMLS lookups for repeated entities."""
    doc = nlp(entity_text)
    # ... entity linking logic ...
    return entity_metadata
```

B.7.2 Parallel Processing

```
import concurrent.futures

def process_documents_parallel(documents, max_workers=4):
    """Process multiple documents in parallel."""
    with concurrent.futures.ThreadPoolExecutor(
        max_workers=max_workers
    ) as executor:
        futures = [
            executor.submit(process_single_document, doc)
            for doc in documents
        ]
        results = [
            future.result()
```

```
            for future in concurrent.futures.as_completed(futures)
        ]
    return results
```

B.8 Reproducibility Checklist

To ensure reproducibility of results:

1. **Data**: Use BioRED dataset (version specified in evaluation)
2. **Random Seeds**: Set all random seeds to 42
3. **Model Versions**: Use exact model versions specified
4. **Hardware**: Document GPU/CPU specifications
5. **Evaluation**: Use provided evaluation scripts
6. **Code Repository**: Complete implementation available at Abioye, 2025

```
# Set random seeds for reproducibility
import random
import numpy as np
import torch

random.seed(42)
np.random.seed(42)
torch.manual_seed(42)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(42)
```

B.9 Troubleshooting Common Issues

B.9.1 Memory Issues

- Reduce batch size for large documents
- Use gradient checkpointing for large models
- Clear cache between batches: `torch.cuda.empty_cache()`

B.9.2 UMLS Licensing

- UMLS access requires registration at https://uts.nlm.nih.gov
- Download UMLS data for SciSpacy: `pip install scispacy-umls`

B.9.3 Neo4j Connection

- Ensure Neo4j service is running: `neo4j start`
- Check connection: `cypher-shell -u neo4j -p password`
- Increase heap memory for large graphs in `neo4j.conf`

Appendix C

Error Analysis and Failure Case Studies

This appendix provides a comprehensive analysis of errors and failure modes observed during the evaluation of our biomedical text extraction pipeline. We examine specific failure cases to understand the limitations of current approaches and identify areas for improvement.

C.1 Overview of System Failures

The evaluation revealed critical failure modes across all components:

- **Entity Recognition**: 70-80% of entities missed at default thresholds
- **Relationship Extraction**: Over 90% of relationships undetected
- **Model Hallucination**: Frequent generation of non-existent entities
- **Output Format Violations**: Consistent failure to follow JSON schemas

Example failure cases and debugging scripts can be found in the repository Abioye, 2025.

C.2 Relationship Extraction Failure Analysis

C.2.1 Quantitative Failure Rates

Table C.1 summarizes the failure rates across all model configurations.

TABLE C.1: Relationship Extraction Failure Rates by Model Configuration

Model	Strategy	Relations Missed	False Positives	Failure Rate
Gemma	Basic	92.7%	92.7%	92.7%
Gemma	Few-shot	91.8%	91.8%	91.8%
Gemma	Structured	93.5%	92.5%	93.0%
MedGemma	Basic	95.0%	91.6%	93.7%
MedGemma	Few-shot	94.1%	91.3%	92.7%
MedGemma	Structured	95.7%	93.2%	94.5%

C.2.2 Common Failure Patterns

Entity Hallucination

Models frequently generated entities not present in the source text:

```
Input entities: ["SCN5A", "long QT syndrome", "arrhythmias"]
Model output: ["SCN5A", "cardiac disease", "heart rhythm disorder"]
```

The model substituted "cardiac disease" for the specific "long QT syndrome" and invented "heart rhythm disorder" as a synonym for "arrhythmias".

Incorrect Entity Matching

```
Ground truth: ("bradycardia", "Na(v)1.5", "Association")
Model output: ("SCN5A", "long QT syndrome", "Association")
```

The model correctly identified related concepts but failed to extract the exact entities mentioned in the text.

Relation Type Confusion

Models struggled to distinguish between similar relation types:

- Association vs. Positive_Correlation: 78% confusion rate
- Drug_Interaction vs. Cotreatment: 65% confusion rate
- Bind vs. Association: 71% confusion rate

C.3 Specific Failure Case Studies

C.3.1 Case Study 1: Complex Medical Relationships

Document ID: 15485686

Input Text:

> "The mutation V1763M in the SCN5A gene encoding Na(v)1.5 causes both bradycardia and tachycardia in patients with long QT syndrome. Treatment with mexiletine and lidocaine showed improvement in arrhythmias."

Expected Relationships (Total: 14):

- (bradycardia, Na(v)1.5, Association)
- (bradycardia, V1763M, Positive_Correlation)
- (tachycardia, Na(v)1.5, Association)
- (arrhythmias, mexiletine, Negative_Correlation)
- ... and 10 more relationships

Model Output (Gemma with few-shot prompting):

- (SCN5A, long QT syndrome, Association) - Partially correct

- (SCN5A, tachycardia, Association) - Missed bradycardia
- Total: 2 out of 14 relationships (14.3% recall)

Failure Analysis:

1. **Entity Resolution**: Model used gene name (SCN5A) instead of protein name (Na(v)1.5)
2. **Missing Relationships**: Failed to identify drug-disease relationships
3. **Relation Type Error**: All relationships labeled as "Association" despite clear causal relationships

C.3.2 Case Study 2: Drug Interaction Scenario

Document ID: 16046395
Input Text:

> "The A118G polymorphism in OPRM1 affects patient response to morphine. Patients with the G allele required higher doses for pain management."

Expected Relationships:

- (A118G, OPRM1, Association)
- (A118G, morphine response, Positive_Correlation)
- (G allele, morphine dose, Positive_Correlation)

Model Failures:

- Generated non-existent entities: "A118" and "G118" separately
- Missed the pharmacogenomic relationship entirely
- No drug-gene interactions identified

C.3.3 Case Study 3: Biochemical Pathway

Document ID: 18457324
Input Text:

> "Carbonyl reductase 3 (CBR3) converts doxorubicin to doxorubicinol in patients receiving anthracycline therapy, potentially contributing to anthracycline-related congestive heart failure."

Model Performance Comparison:

TABLE C.2: Model Output Comparison for Biochemical Pathway Extraction

Model	Extracted Relationships
Gemma (Basic)	(carbonyl reductase 3, anthracyclines, Cotreatment) (patients, anthracyclines, Cotreatment)
MedGemma (Few-shot)	(CBR3, doxorubicin, Association) (anthracycline, heart failure, Association)
Ground Truth	(carbonyl reductase 3, doxorubicin, Conversion) (carbonyl reductase 3, doxorubicinol, Conversion) (doxorubicinol, congestive heart failure, Positive_Correlation) ... (6 more relationships)

C.4 Prompt Strategy Effectiveness Analysis

C.4.1 Basic Prompting Failures

Common Issues:

- Output not in requested JSON format (31% of responses)
- Mixed entity types and relations in output
- Incomplete triplets missing one or more components

Example Failure:

```
Expected format: [{"entity1": "...", "entity2": "...", "relation": "..."}]
Actual output: "SCN5A is associated with heart conditions including
                arrhythmias and long QT syndrome"
```

C.4.2 Few-Shot Prompting Failures

Despite providing examples, models failed to generalize:

- Copied example relationships verbatim (12% of cases)
- Misapplied example patterns to unrelated contexts
- Slight improvement in format compliance (+8%) but not accuracy

C.4.3 Structured JSON Prompting Failures

Schema Violations:

```
Schema requirement: {"entity1": str, "entity2": str, "relation": str}
Common violations:
- Added extra fields: "confidence", "source", "context"
- Nested objects instead of flat structure
- Arrays where strings expected
```

C.5 Domain-Specific Model Underperformance

C.5.1 MedGemma vs Gemma Comparison

Contrary to expectations, MedGemma performed worse than the general Gemma model:

TABLE C.3: Comparative Error Analysis: MedGemma vs Gemma

Error Type	MedGemma Rate	Gemma Rate
Entity hallucination	48.3%	41.2%
Format violations	38.7%	32.1%
Relation type errors	71.2%	65.8%
Complete failures	15.3%	11.7%

C.5.2 Hypotheses for Domain Model Underperformance

1. **Overfitting to Training Domain**: MedGemma may be overfitted to specific medical text formats not matching BioRED abstracts
2. **Reduced General Reasoning**: Domain specialization potentially reduced general language understanding capabilities
3. **Prompt Sensitivity**: Medical models may require different prompting strategies than evaluated
4. **Task Mismatch**: Pre-training focused on different biomedical tasks than relation extraction

C.6 Error Cascading Effects

C.6.1 NER to Relationship Extraction

Errors in entity recognition cascaded to relationship extraction:

- Missed entities: 100% relationship extraction failure
- Incorrect boundaries: 87% relationship misalignment
- Wrong entity types: 73% relation type errors

C.6.2 Example of Error Propagation

```
Step 1 - NER Output:
  Expected: ["carbonyl reductase 3", "doxorubicin", "doxorubicinol"]
  Actual: ["carbonyl reductase", "doxorubicin", "heart failure"]

Step 2 - Relationship Extraction:
  Cannot find: (carbonyl reductase 3, doxorubicinol, Conversion)
  Hallucinated: (carbonyl reductase, heart failure, Causes)
```

C.7 Computational Error Analysis

C.7.1 Memory and Resource Failures

- Out-of-memory errors for documents > 4000 tokens
- Timeout failures for complex relationship graphs
- GPU memory exhaustion with batch size > 16

C.7.2 Performance Degradation Patterns

TABLE C.4: Error Rates by Document Complexity

Document Length	Entities	Relations	Error Rate
< 500 tokens	< 10	< 5	85.2%
500-1000 tokens	10-20	5-15	91.7%
1000-2000 tokens	20-40	15-30	94.3%
> 2000 tokens	> 40	> 30	97.8%

C.8 Recommendations Based on Error Analysis

C.8.1 Immediate Improvements

1. **Entity Validation**: Implement strict validation ensuring extracted entities exist in source text
2. **Output Parsing**: Add robust JSON parsing with fallback strategies
3. **Confidence Thresholds**: Implement relationship confidence scoring

C.8.2 Architectural Changes

1. **Pipeline Redesign**: Consider joint entity-relation extraction models
2. **Model Selection**: Use larger language models (GPT-4, Claude) for relationship extraction
3. **Hybrid Approaches**: Combine rule-based methods for high-precision scenarios

C.8.3 Training and Fine-tuning

1. **Task-Specific Training**: Fine-tune models specifically on BioRED-style relation extraction
2. **Prompt Optimization**: Develop biomedical-specific prompting strategies
3. **Error-Aware Training**: Include common error patterns in training data

C.9 Conclusion

The error analysis reveals fundamental limitations in current small language model approaches for biomedical relationship extraction. The 90%+ failure rate indicates that:

- Small models lack sufficient capacity for complex biomedical reasoning
- Domain-specific pre-training alone is insufficient without task-specific fine-tuning
- The complexity of biomedical relationships exceeds current model capabilities
- Alternative approaches or significantly larger models are required for practical applications

These findings highlight the need for continued research in biomedical NLP, particularly in developing more capable models and robust extraction pipelines for clinical applications.

Bibliography

Abioye, Oyetunji Daniel (2025). *Github Repository*. URL: https://github.com/Tunji17/masters-thesis.

Beltagy, Iz, Kyle Lo, and Arman Cohan (2019). *SciBERT: A Pretrained Language Model for Scientific Text*. arXiv: 1903.10676 [cs.CL]. URL: https://arxiv.org/abs/1903.10676.

Bolton, Elliot et al. (2024). *BioMedLM: A 2.7B Parameter Language Model Trained On Biomedical Text*. arXiv: 2403.18421 [cs.CL]. URL: https://arxiv.org/abs/2403.18421.

Cowell, L. and B. Smith (2020). "Ontology and the future of the data-rich medical record". In: *Stud Health Technol Inform* 270, pp. 145–149.

Google AI (2025). *Gemma: Open-Source Foundation Models*. URL: https://deepmind.google/models/gemma/gemma-3.

Google DeepMind (2025). *MedGemma: A Gemma 3 variant optimized for medical text and image comprehension*. URL: https://deepmind.google/models/gemma/medgemma.

Hier, D. B. et al. (2025). *Efficient standardization of clinical notes using large language models*. arXiv: 2501.00644. URL: https://arxiv.org/html/2501.00644v1.

Himmelstein, D. S. et al. (2017). "Systematic integration of biomedical knowledge prioritizes drugs for repurposing". In: *eLife* 6, e26726. DOI: 10.7554/eLife.26726.

Huang, Kexin, Jaan Altosaar, and Rajesh Ranganath (2020). *ClinicalBERT: Modeling Clinical Notes and Predicting Hospital Readmission*. arXiv: 1904.05342 [cs.CL]. URL: https://arxiv.org/abs/1904.05342.

Ji, Z. et al. (2023). *Survey of Hallucination in Natural Language Generation*. arXiv: 2202.03629.

Klug, K. et al. (2024). "From admission to discharge: a systematic review of clinical natural language processing along the patient journey". In: *BMC Medical Informatics and Decision Making* 24.238, pp. 1–13. DOI: 10.1186/s12911-024-02641-w.

Kong, H.-J. (2019). "Managing unstructured big data in healthcare system". In: *Healthcare Informatics Research* 25.1, pp. 1–2. DOI: 10.4258/hir.2019.25.1.1.

Laskar, M. T. R. et al. (2025). *Improving Automatic Evaluation of Large Language Models in Biomedical Relation Extraction via LLMs-as-the-Judge*. Accepted at ACL 2025. arXiv: 2506.00777.

Leaman, R., R. Islamaj Dogan, and Z. Lu (2013). "DNorm: disease name normalization with pairwise learning to rank". In: *Bioinformatics* 29.22, pp. 2909–2917. DOI: 10.1093/bioinformatics/btt474.

Leaman, R. and Z. Lu (2016). "TaggerOne: joint named entity recognition and normalization with semi-Markov Models". In: *Bioinformatics* 32.18, pp. 2839–2846. DOI: 10.1093/bioinformatics/btw343.

Lee, Jinhyuk et al. (Sept. 2019). "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4. Ed. by Jonathan Wren, 1234–1240. ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btz682. URL: http://dx.doi.org/10.1093/bioinformatics/btz682.

Liu, L. et al. (2024). *A survey on medical large language models: Technology, application, trustworthiness, and future directions*. arXiv: 2406.03712. URL: https://arxiv.org/html/2406.03712v2.

Luo, Ling et al. (July 2022). "BioRED: a rich biomedical relation extraction dataset". In: *Briefings in Bioinformatics* 23.5, bbac282. ISSN: 1477-4054. DOI: 10.1093/bib/bbac282. eprint: https://academic.oup.com/bib/article-pdf/23/5/bbac282/45936115/bbac282.pdf. URL: https://doi.org/10.1093/bib/bbac282.

McKerns, M., M. Aivazis, and M. Scherer (2021). *Pathos: Parallel and Distributed Computing Framework for Python*. URL: https://pathos.readthedocs.io.

Milvus (2025). *What are the use cases for knowledge graphs in healthcare?* AI Quick Reference Guide. URL: https://milvus.io/ai-quick-reference/what-are-the-use-cases-for-knowledge-graphs-in-healthcare.

Neo4j, Inc. (2023). *Cypher Query Language Documentation*. URL: https://neo4j.com/docs/cypher-refcard/current/.

Neumann, M. et al. (2019). "ScispaCy: Fast and Robust Models for Biomedical Natural Language Processing". In: *Proceedings of the 18th BioNLP Workshop*. Association for Computational Linguistics, pp. 319–327. DOI: 10.18653/v1/W19-5034. URL: https://github.com/allenai/scispacy.

Phan, M. C., A. Sun, and Y. Tay (2019). "Robust Representation Learning of Biomedical Names". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 3275–3285. DOI: 10.18653/v1/P19-1317.

Reynolds, L. and K. McDonell (2021). *Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm*. arXiv: 2102.07350.

Rotmensch, M. et al. (2017). "Learning a health knowledge graph from electronic medical records". In: *Scientific Reports* 7, p. 5994. DOI: 10.1038/s41598-017-05778-z.

Schwartz, A. S. and M. A. Hearst (2003). "A simple algorithm for identifying abbreviation definitions in biomedical text". In: *Pacific Symposium on Biocomputing*, pp. 451–462.

Singhal, K. et al. (2022). *Large Language Models Encode Clinical Knowledge*. arXiv: 2212.13138.

Sung, M. et al. (2020). "Biomedical Entity Representations with Synonym Marginalization". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 11–20. DOI: 10.48550/arXiv.2005.00239.

Sänger, M. et al. (2024). *HunFlair2 in a cross-corpus evaluation of biomedical named entity recognition and normalization tools*. arXiv: 2402.12372 [cs.CL]. URL: https://arxiv.org/abs/2402.12372.

U.S. National Library of Medicine (2024). *Unified Medical Language System (UMLS) Metathesaurus*. URL: https://www.nlm.nih.gov/research/umls/.

Yazdani, A., I. Stepanov, and D. Teodoro (2025). *GLiNER-BioMed: A Suite of Efficient Models for Open Biomedical Named Entity Recognition*. arXiv: 2504.00676.

Zimbres, R. (2024). "Building Knowledge Graphs from Scratch Using Neo4j and Vertex AI". In: *Medium*, pp. 85–107. URL: https://medium.com/@rubenszimbres/building-knowledge-graphs-from-scratch-using-neo4j-and-vertex-ai-8311eb69a472.