People's Democratic Republic of Algeria
Ministry of Higher Education and Scientific Research

Houari Boumediene University of Science and Technology
Faculty of Computer Science
Department of Artificial Intelligence and Data Science (IASD)

Master's thesis

Branch: Computer Science
Specialty: BIOINFO

Intelligent Drug-Drug Interaction Extraction: A Natural Language Processing Approach to Biomedical Text Analysis

---

Topic proposed by
Mrs.KALI ALI Selma
Mr.Khennak Ilyes
Member of the Jury
Mr.BESSA Brahim
Mrs.AMEUR Imane

Presented By :
ALLALI Ryad
ALLALI Rayan
Defended On
-/-/2025

Project:`BIOINFO02`

ACKNOWLEDGEMENTS

First and foremost, we would like to send our greatest gratitude to Allah, the Almighty, for everything and every blessing, especially the determination, patience, and passion that helped us throughout our academic journey.

This work would not have been completed without the immense support from many remarkable parties. Therefore, we would like to express our sincere gratitude to those who have shaped our academic journey.

We would like to thank our supervisors, Miss Kali Ali and Mr. Khennak, for giving us the chance and opportunity to work on this interesting project. but most importantly, We would also like to sincerely and especially thank Miss Kali for her support and guidance throughout the project. She made the whole process easier and less stressful. We could never thank her enough. Her reviews, important notes, suggestions, and even lending us a machine to conduct the experiments and tests played a major role in the success of this work. Thank you, Miss!

We are also grateful and thankful to the jury members, the president, Mr.BESSA Brahim, and Miss AMEUR Imane, for dedicating their valuable time to reviewing our thesis.

We would also like to thank everyone who helped us and had a big or small role in this work, including, but not limited to, our friends Sohaib Zouambia and Mossab Badla, especially. We are also grateful to all the teachers and instructors of the "Informatique" Department who contributed to our education.

Finally, we would like to thank our family and especially our father and mother for the support, sacrifices, and hard work they invested in our education from primary school up to university and for everything else.

ABSTRACT

Biomedical literature relation extraction is of critical significance in transforming unstructured text into structured knowledge, particularly for Drug-Drug Interaction (DDI) extraction. In this thesis, we introduce a novel hybrid model for DDI extraction that addresses both Drug Named Entity Recognition (DNER) and relation classification by using information-rich sources. For DNER, we employ BioBERT and fine-tune it to locate drug mentions in text accurately. For relation classification, we integrate BioBERT contextual semantic embeddings with syntactic structures obtained through dependency parsing, which generates graph representations of grammatical relations between tokens. We feed these graphs into a Graph Attention Network (GAT) whose nodes are initialized by a mixture of BioBERT token embeddings and Word2Vec embeddings. To effectively integrate features, we employ two fusion strategies that align the BioBERT [CLS] embedding with the GAT output: a conventional fusion approach and a cross-attention mechanism. To counter class imbalance common in biomedical corpora, we apply negative filtering and focal loss. By utilizing dependency-parsed graphs and multimodal feature fusion, this work emphasizes the importance of integrating semantic, syntactic, and lexical information, laying a strong foundation for the advancement of biomedical relation extraction.

ABBREVIATIONS

DDI : Drug Drug Interaction
ADR : Adverse Drug Reactions
NLP : Natural Language Processing
LSTM:Long Short-Term Memory
GRU:Gated Recurrent Unit
POS :Part of Speech Tagging
NER : Named Entity Recognition
Bi-lSTM:Bidirectional LSTM
IR : Information retrieval
RNN :recurrent neural network
CNN: Convolutional neural network
BERT:Bidirectional Encoder Representations from Transformers
MLM :Masked LM
NSP: Next Sentence Prediction
CBOW:Continuous Bag of Words
RE : Relation Extraction
IE:Information Extraction
OOV:Out-Of-Vocabulary
QA: Question Answering
TF-IDF:Term Frequency-Inverse Document Frequency
GloVe:Global Vector for word representation
GCN: Graph Convolutional network
SPLs:structured product labels
GNN:Graph neural networks
GAT:Graph Attention Networks
AI: Artificial Intelligence
seq2seq:sequence-to-sequence
REGEX:regular expressionS
NN : Neural Networks
DNER: Drug Entity Recognition
DL: Deep Learning
LLM: Large Language Models
AUC: Area Under the Curve
ML: Machine Learning
SVM:support vector machine

CONTENTS

General Introduction **1**

1 State of the Art **3**
1 Introduction . . . 3
2 DDI Extraction . . . 3
2.1 Definition . . . 3
2.2 Available Data Sources and their Challenges . . . 4
2.3 Clinical Impact of Automated DDI Extraction . . . 4
3 NLP in Biomedical . . . 4
3.1 Peculiarities of Medical Language . . . 4
3.2 Role of NLP in Biomedicine . . . 6
4 Analysis of Existing Methods for DDI Extraction . . . 7
4.1 Existing Methods . . . 7
4.1.1 Rule-Based Methods . . . 7
4.1.2 Statistical Machine Learning-Based Methods . . . 7
4.1.3 Deep Learning-Based Methods . . . 8
4.1.4 Hybrid-Based Methods . . . 9
4.2 Comparison of DDI Extraction Methods . . . 9
5 Conclusion . . . 10

2 Theoretical Background **11**
1 Introduction . . . 11
2 Fundamentals of NLP . . . 11
2.1 NLP Concepts . . . 11
2.1.1 Tokenization . . . 11
2.1.2 Part-Of-Speech Taging . . . 12
2.1.3 Dependency Parsing . . . 12
2.2 Word Representation . . . 13
2.2.1 Word Encoding . . . 13
2.2.2 Word Embeddings . . . 14
3 DL for NLP . . . 15
3.1 Transformers . . . 15
3.1.1 Key Concepts . . . 15
3.1.2 Encoder-Decoder Structure . . . 17

3.2 Pre-trained Large Language Models for Feature Extraction 18
3.2.1 BERT 19
3.2.2 Domain-Specific Variants of BERT 21
3.3 Fine-tuning for LLMS 21
3.4 Graph Neural Networks 21
3.4.1 Graph Structures 21
3.4.2 Overview of GNNs 22
4 Evaluation of Automatic DDI Extraction Models 24
4.1 Evaluation Metrics 24
4.2 Benchmarks and Available Datasets 25
4.2.1 DDI Extraction 2013 25
4.2.2 TAC 2018 and TAC 2019 DDI Extraction 26
5 Conclusion 26

3 Project Design and Methodology **27**
1 Introduction 27
2 Data Selection 27
2.1 Data Description 27
2.2 Data Structure 27
3 Overview of the Proposed Solution 29
4 Drug Name Entity Recognition Model 30
4.1 Preprocessing Pipeline 30
4.1.1 Data Cleaning 31
4.1.2 BIO Tagging 31
4.1.3 Tokenization and Label Alignment 31
4.2 Model Architecture 31
5 Drugs Relation Extraction Model 32
5.1 Preprocessing Pipeline 33
5.1.1 Data Cleaning 34
5.1.2 Drug Blinding 34
5.1.3 Sentence Multiplication and Drug Tagging 34
5.1.4 Imbalances Dataset 35
5.2 Model Architecture 36
5.2.1 Building the Dependency Parsing Graph 37
5.2.2 GAT Network 39
5.2.3 Feature Fusion 39
5.2.4 Classification 40
6 Complete DDI Extraction Workflow 40
7 Conclusion 41

4 Experiments and Results **43**
1 Introduction 43
2 Experimental Environment 43
2.1 Hardware Specification 43
2.2 Software Specification 43
3 Evaluation and Analysis 45
3.1 DNER Evaluation 45
3.1.1 Hyperparameter Tuning 46

3.1.2 Model Results 46
3.2 RE Evaluation 48
3.2.1 Handling Class Imbalance 48
3.2.2 Graph-Based Representations 50
3.2.3 Feature Ablation and Analysis 51
3.2.4 Hyperparameter Tuning 53
3.2.5 Ensemble Learning 56
3.2.6 Overfitting Analysis 56
3.2.7 Model Results 58
3.3 End-to-End Architecture Performance 60
3.3.1 Instance-Level Evaluation 61
3.3.2 Relation Classification on Extracted Instances 61
4 Application Features and User Interface 63
5 Conclusion 66

General Conclusion **67**

A THEORETICAL BACKGROUND **69**
1 POS Tagging Categories 69
2 TF-IDF and Bag-of-Words 70

Bibliography **79**

LIST OF TABLES

1.1 Examples of medical terms and their linguistic components. 5
1.2 Common Medical Abbreviations 5
1.4 Comparison of various methods for text processing 10

2.1 Types of tokenization 12
2.2 One-Hot Encoding Example for Color Categories 13
2.3 Different sizes of BERT models. 19
2.4 Statistics of DDI Extraction 2013 Dataset 26
2.5 Statistics for TAC 2018 and TAC 2019 datasets 26

3.1 Example of BIO Tagging for DNER 31
3.2 Data statistics before and after negative filtering 36

4.2 Sample comparison of BERT-based models for DNER 45
4.3 Effect of stop word removal on BioBERT performance 45
4.4 Best hyperparameters found for DNER 46
4.5 Performance comparison between baseline and Optuna-tuned configuration 46
4.6 Evaluation metrics for the DNER model 46
4.7 Sample comparison of BERT-based models for RE 48
4.8 Impact of Negative Filtering on macro F1-scores for DDI Extraction . . . 48
4.9 Comparison of F1 Scores with and without Augmentation 49
4.10 Macro F1 comparison across α_t strategies. 50
4.11 Comparison of Graph Representations 51
4.12 Comparison between fusion strategies 52
4.13 Performance comparison for different feature fusion combinations. 52
4.14 Best Hyperparameters Hyperparameter Settings 55
4.15 Performance comparison between baseline and Optuna-tuned configuration 56
4.16 Comparison between the Single Best Model and Ensemble Learning (Hard Voting) 56
4.17 Performance comparison of BioBERT and proposed methods. 59
4.18 Evaluation metrics per class and global macro average 60
4.19 Instance-level evaluation metrics for the full pipeline 61
4.20 Evaluation metrics per class and global macro average 62

A.1 List of Universal POS Tags 69

LIST OF FIGURES

2.1 Example of dependency parsing 13
2.2 Word2Vec embeddings 14
2.3 Multi-head self-attention mechanism 17
2.4 The architecture of the Transformer 18

3.1 DDI EXTRACTION 2013 DOCUMENT FORMAT 28
3.2 Example of a DDI extraction task 30
3.3 Overview of the proposed two-stage architecture for DDI extraction . . . 30
3.4 DNER Preprocess Pipline 31
3.5 DNER model architecture 32
3.6 Dependency tree illustrating syntactic structure linking drug mentions. . 33
3.7 Drug RE Preprocess Pipeline 34
3.8 Architecture of the proposed Model for DDI relation extraction 38
3.9 Cross-attention mechanism using GAT node features as queries and BioBERT `[CLS]` as keys and values 40
3.10 Full pipeline for DDI extraction. 42

4.1 Confusion matrix for the DNER model 47
4.2 Class-wise AUC scores for the DNER model 47
4.3 Parallel coordinate plot from Optuna showing the influence of key hyperparameters 53
4.4 Objective values across Optuna trial 54
4.5 Training and Validation loss before regularization 57
4.6 Training and validation macro F1-score before regularization 57
4.7 Training and validation loss for our proposed model 58
4.8 Macro F1-score for our proposed model 58
4.9 RE Confusion Matrix 60
4.10 Full Pipeline Confusion Matrix 62
4.11 welcome Page 63
4.12 Statistics and dashboard 64
4.13 DDI Analyser Page 65
4.14 DDI Analyser Page (DDI Extracted) 65
4.15 DDI Analyser Page (Drug Entities Extracted) 66

GENERAL INTRODUCTION

A Drug-Drug Interaction (DDI) is a reaction between two (or more) drugs. It occurs when one drug influences the level or activity of another drug. This can affect how a drug works or cause unwanted side effects. DDIs pose a serious health issue as they are one of the main causes of Adverse Drug Reactions (ADRs) [11]; they can cause some serious unexpected ADRs that can be life-threatening. A study from the USA reported that DDIs were the cause of 57% of ADR-related admissions and 4.3% of all hospital admissions [50]. In a study conducted in Saudi Arabia on 310 geriatric patients, the overall prevalence of DDIs was 90.64%, with 68.38% representing potentially harmful interactions. [7]. In a similar study conducted on 313 patients here in Algeria, they found that the overall prevalence of DDIs was estimated at 90.7%, with an average of 4.7 interactions per patient [23].

With the increase in chronic conditions such as cancer and diabetes, coupled with advances in medical treatment, the prevalence of polypharmacy has significantly risen, especially among the elderly population. Older adults often suffer from multiple comorbidities and, as a result, are exposed to multiple drugs; polypharmacy is commonly defined as the regular use of five or more medications simultaneously. A study on polypharmacy in Egypt found that the prevalence of polypharmacy among older adults was 85.3%, which is similar to the results of previous studies conducted in Mexico (84.5%), Abu Dhabi (89%), and Saudi Arabia (89%) [46].

The use of multiple medications can trigger the incidence of DDI. The risk of DDI is reported to escalate linearly as the number of drugs consumed increases [51]. Consuming five to seven drugs and ten to fourteen drugs increased the risk of potentially clinically relevant DDI by about 20% to 30% and 40% to 60%, respectively [25]. This means that healthcare professionals need to be extra careful when diagnosing patients and prescribing treatments, especially in cases involving polypharmacy.

Although there is a large number of drug databases and semistructured resources (such as MedLine, DrugBank, and Drug Interactions Facts) that they can browse, the problem is that, especially with the rapid advancements in the medical field, large datasets and articles are being published regularly about newly discovered DDIs. MedLine alone sees an increase between 10,000 and 20,000 published research articles per week [26]. Therefore, researchers and healthcare professionals must review many drug safety reports and publications in medicine and pharmacology to keep up to date with the latest findings on DDI. This is why there is an urgent need for DDI extraction using Natural language processing.

In this project, we focus on extracting pairwise DDIs from biomedical articles. Although some studies have explored DDI extraction or prediction involving multiple drugs, our work explicitly targets the extraction of interactions between two drugs at a time. This focus is motivated by the lack of high-quality annotated datasets for multi-drug DDI extraction and the limited amount of research conducted on this topic. We developed two models: one for extracting drug entities and another for identifying interaction relations between drug pairs. The work is organized into four chapters:

- **Chapter 1:** This chapter introduces the concept of DDI extraction, discusses its clinical significance, and outlines the various data sources available. It also reviews state-of-the-art methods for DDI extraction, highlighting their limitations and the open challenges in the field.
- **Chapter 2:** This chapter presents the theoretical background necessary to understand the core concepts of our work.
- **Chapter 3:** This chapter details the selected dataset and our proposed solution, including system architecture and data preprocessing steps
- **Chapter 4:** The final chapter presents the experimental results, performance evaluation, and a demonstration of the developed application interface.

CHAPTER 1

STATE OF THE ART

1 Introduction

In the field of pharmacovigilance, Drug-Drug Interactions (DDIs) pose a serious health issue as they are one of the main causes of Adverse Drug Reactions (ADRs) [11]. When two or more drugs interact, they can cause some severe, unexpected ADRs that can be life-threatening [50, 23, 7, 46].

Health professionals and experts must exercise heightened caution when diagnosing and prescribing drugs. Although there is a large number of drug databases and semi-structured resources (such as MedLine, DrugBank, and Drug Interactions Facts) that they can browse, the problem is that, especially with the rapid advancements in the medical field, large datasets and articles are being published regularly about newly discovered DDIs. MedLine alone sees an increase of 10,000 to 20,000 published research articles each week [26]. Therefore, researchers and healthcare professionals must review many drug safety reports and publications in medicine and pharmacology to keep up to date with the latest findings on DDI. This highlights the critical need for automated DDI extraction using Natural Language Processing (NLP) techniques.

In this chapter, we will provide a comprehensive review of DDI extraction. We will begin by introducing the concept of drug-drug interactions, discussing their clinical significance, and the various data sources available. Next, we will examine the role of NLP in biomedical text analysis. Finally, we will discuss state-of-the-art methods for DDI extraction while also addressing the limitations and open challenges in the field.

2 DDI Extraction

2.1 Definition

A DDI is a reaction between two (or more) drugs. It occurs when one drug influences the level or activity of another drug. This can affect how a drug works or cause unwanted side effects. Research in this field can be split into three types: detection, prediction, and extraction. Detection methods identify known DDIs in structured or unstructured data, while prediction techniques help researchers discover potential new DDIs. On the other hand, DDI extraction focuses on extracting existing DDI information from the biomedical

literature. It involves identifying and extracting documented interactions between drugs from unstructured text. The process is divided into two main steps:

1. **Drugs Recognition:** Given a text or report written by health professionals or experts, the system detects the drugs mentioned in the document.

2. **Interaction Extraction:** After recognizing the drugs, it determines the type of relation between the mentioned drugs, if there is any.

Example: given the sentence "Warfarin may cause bleeding when taken with Aspirin," a DDI extraction system would typically extract:

- **Drug 1**: Warfarin
- **Drug 2**: Aspirin
- **Interaction**: risk of bleeding.

2.2 Available Data Sources and their Challenges

Currently, multiple DDI databases, such as Drug Interactions Facts, Stockley, DailyMed, WebMD, and DrugBank, are accessible to researchers, physicians, and patients. However, most DDI databases store DDIs as textual descriptions rather than structured relational data, making direct access difficult for healthcare professionals.

Moreover, these databases are not updated regularly, with some having update cycles as long as three years [52]. This delay means that newly discovered interactions may not be immediately available to healthcare professionals, potentially leading to outdated prescriptions. As a result, many DDIs remain hidden in unstructured biomedical literature despite their clinical significance. Given the rapid growth of biomedical research, most newly discovered DDIs are first reported in clinical pharmacology journals and technical reports, making medical literature the most effective source for DDI detection.

2.3 Clinical Impact of Automated DDI Extraction

To avoid DDIs during prescription, experts and healthcare professionals must continuously monitor the latest findings by reviewing the large volume of published articles and manually extracting reported DDIs, which is an almost impossible task. Therefore, DDI extraction, which automatically detects DDIs in unstructured text and classifies them into predefined categories, has become increasingly important in medical text mining. It can be highly beneficial to the pharmaceutical industry, allowing the identification and extraction of relevant DDI information while significantly reducing the time healthcare professionals spend reviewing the literature. Moreover, developing automated DDI extraction tools is essential for improving and updating drug knowledge databases.

3 NLP in Biomedical

3.1 Peculiarities of Medical Language

Medical language is the language used by medical experts in their professional communication and incorporates more than 2,500 years of development influenced mostly by

Greek and Latin medical traditions [39]. Medical language is precise and difficult primarily because it varies greatly from ordinary language in several significant aspects. First, medical language favors clarity and accuracy over simplicity and uses technical terminology instead of common words. Examples of common medical terms, defined and deconstructed into their parts, are presented in the Table 1.1.

Table 1.1: Examples of medical terms and their linguistic components.

Term	**Definition**	**Breakdown**
Pharyngitis	Sore throat	*pharyng-* = pharynx, or throat *-itis* = disease or inflammation
Ante cibum (AC)	Before meals	*ante* = before *cibum* = food
Intravenous	administered through a vein	*intra-* = within *-venous* = relating to a vein
Myocardial infarction	Heart attack	*myo-* = muscle *-cardial* = of the heart *infarction* = tissue death

In addition to technical terms, medical professionals frequently use a wide range of abbreviations. These shorthand notations are widely understood within the medical community but can be difficult for non-specialists to interpret [64]. Table 1.2 illustrates some common examples and their meaning.

Table 1.2: Common Medical Abbreviations

Abbreviation	**Meaning**
CVS	Cardiovascular System
CNS	Central Nervous System
MSS	Muscular Skeletal System
GIT	Gastrointestinal Tract
NG Tube	Nasogastric Tube
CBC	Complete Blood Count
SOB	Shortness of Breath

Additionally, a unique feature of medical language is the use of eponyms—terms. Eponyms are words that are derived from the names of people who are closely associated with the product, theory, idea, or service. These names might be mythological names or actual names of people that exist(ed). Some well-known examples include:

- **Adam's Apple** – Medically known as the laryngeal prominence, this term originates from the biblical story of Adam and the forbidden fruit.
- **Alzheimer's Disease** – A neurodegenerative disorder first described by German psychiatrist and neuropathologist Alois Alzheimer.
- **McArdle's Disease** – A glycogen storage disorder identified by Dr. Brian McArdle.
- **Cushing's Syndrome** – A condition caused by prolonged exposure to high cortisol levels, first described by Harvey Cushing.

- **Broca's Area** – The brain region responsible for speech production, named after Pierre Paul Broca.
- **Babe-Ernst Bodies** – Granules discovered by Romanian bacteriologist Victor Babe and German pathologist Ernst Paul.

In addition to these challenges, biomedical literature is filled with synonyms and highly context-dependent phrases, adding another layer of complexity. Furthermore, naming conventions in the medical field are often more intricate than those in natural language. In many cases, a single name consists of multiple words, for example, "hereditary non-polyposis colorectal cancer syndrome." Additionally, the same entity can have multiple variant forms. For instance, "zolmitriptan", "Zomig," and "Zomigon" all refer to the same drug.

3.2 Role of NLP in Biomedicine

NLP is a subfield of artificial intelligence (AI) that enables computers to understand, interpret, and generate human language. In biomedicine, NLP plays a crucial role in analyzing the vast and rapidly growing body of biomedical literature, which includes scientific articles, clinical notes, and other unstructured data collectively representing more than 80% of all digital data. Given the immense volume of online information, NLP algorithms and techniques help address medical language challenges by extracting meaning and context from textual data, facilitating clinical decision-making, predictive analytics, and medical research [58]. NLP has a wide range of applications in biomedicine, including:

- **Text Classification**: text classification is categorizing texts into predefined classes, for instance, classifying patient records based on disease types [43] or sorting research articles by specific medical topics.
- **Question Answering (QA)**: Biomedical QA systems provide precise answers to advanced users' queries.
- **Text Summarization**: aims to generate a condensed version of a longer text while retaining its essential meaning and information. It's often used for summarizing news articles or academic papers for easier consumption [71].
- **Named Entity Recognition (NER)**: NER is a subtask of information extraction that focuses on identifying and categorizing specific entities in text, such as nouns. It involves automatically scanning unstructured text to locate "entities" for term normalization and classification into predefined categories, such as person names or organizations [71]. In the biomedical domain, NER is used to extract key biomedical entities, including diseases, drugs, genes, effects, and symptoms, from text.
- **Relation Extraction (RE)**: RE is the process of identifying and classifying relationships between entities in text. For example, in our context, it can be used to identify the effects of DDI [19].

4 Analysis of Existing Methods for DDI Extraction

In this section, we will present existing methods for DDI extraction, highlight their limitations, and compare them.

4.1 Existing Methods

In recent years, considerable efforts have been devoted to DDI extraction; existing methods can be divided into four categories, as described below.

4.1.1 Rule-Based Methods

Rule-based approaches define a set of rules for possible textual relationships, known as patterns, that capture similar structures used to express relationships. They are designed to emulate the decision-making abilities of human experts. These approaches rely on a set of "if-then" rules to derive actions from given conditions [38].

In the context of DDI extraction, a pharmacist or domain expert defines a set of domain-specific lexical patterns to capture the most common expressions of DDIs in texts. For instance, the authors in [52] utilize linguistic rules. These lexical patterns are then matched with textual data to identify and extract DDI.

4.1.2 Statistical Machine Learning-Based Methods

Statistical Machine Learning (ML) methods extract DDIs based on patterns/features in data (both structured and unstructured) that augment automatic extraction. Unlike Rule-based methods, which depend upon a set of pre-defined rules and expert input, adding more manual work to the process ML-based methods treat DDI extraction as a standard supervised learning problem over annotated corpora. For that, ML-based methods usually show better performance and better portability. There are mainly two types of machine learning systems [69]:

- **Feature-Based**, representing each data instance as a feature vector. These methods typically rely on feature engineering, where relevant linguistic, syntactic, or pharmacological information is extracted from data and used as input for learning algorithms. Some of the most commonly used approaches include Support Vector Machines (SVM) [14].
- **Kernel-Based Methods**, which exploit structural representations of data instances to compute similarities between them without explicitly transforming them into feature vectors [41]. SVMs, a common implementation of these methods, leverage kernel functions to map input data into higher-dimensional spaces, enabling effective classification of complex relationships such as drug-drug interactions [18].

The problem with these methods is that they heavily rely on tedious feature engineering and redundant feature selection, and defining the feature set in a supervised manner will also limit the identification of other valuable patterns. Moreover, as these methods are based on traditional ML models and are not capable of extracting deep features from input data, they will become much less effective when dealing with large data sets [31].

4.1.3 Deep Learning-Based Methods

In recent years, Deep Learning (DL) has become the primary approach to extracting DDI, significantly surpassing the performance of traditional ML methods. This shift can largely be attributed to DL's ability to eliminate the extensive feature engineering process, which often demands significant effort, by automatically learning representations from data through a hierarchical methodology. Here are some examples of DL-based approaches:

- **Convolutional Neural Network Based Approaches**:

  Convolutional Neural Network (CNN) is an artificial neural network designed initially to process structured grid data, such as images. However, CNNs have been effectively adapted for text classification and relation extraction tasks, including DDI extraction. In text processing, CNNs operate on distributed word representations, using convolutional filters to identify important local syntactic and semantic patterns in the data. Unlike traditional ML methods such as SVMs, CNNs can automatically learn feature representations from raw text [16]. In the context of DDI extraction, multiple implementations of CNNs have been proposed [80, 77].

- **Recurrent Neural Network Based Approaches**:

  Unlike CNNs, which may struggle with sequential data and long-range dependencies, Recurrent Neural Networks (RNNs) are specifically designed to handle such data; RNNs process sequential data using loops that can recall and detect patterns in those sequences.

  Traditional RNN has exploded and vanishing gradient problems [45]. To address these problems, Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU) with their gate mechanisms can control the extent of what is to be forgotten given the input, the previous state, and the current state. These networks can explicitly learn when to forget information and when to store it, which is suitable for an input of arbitrary length. Undoubtedly, these variants efficiently extract DDIs [16]. Several implementations of RNN-based models have been proposed for DDI extraction [34].

- **Attention-Based Approaches**: Unlike traditional RNNs, which process sequences step-by-step and may struggle with capturing very long-range dependencies, attention mechanisms enable models to dynamically weigh the importance of each part of the input when making predictions. This dynamic weighting allows the model to focus on the most relevant information, enhancing its ability to understand complex relationships within the data [4]. Attention mechanisms help the model understand how much each input should pay attention to other inputs. They are all about understanding the context and using it to update the meaning of every word in a sequence. In the context of DDI extraction, attention mechanisms have been effectively employed to enhance performance by concentrating computational resources on the most relevant parts of the text containing drug interaction information. Various approaches incorporating attention mechanisms have been proposed, often in combination with RNN [81].

- **Transformer-Based Approaches**: In recent years, transformer-based models have significantly advanced the field of DDI extraction. These models utilize self-attention mechanisms to capture complex contextual relationships within the text,

enabling the identification of intricate interactions between drugs. These models effectively understand domain-specific language nuances by pre-training on extensive biomedical corpora, improving performance in DDI extraction tasks. They have demonstrated superior accuracy in extracting DDIs from biomedical literature, achieving state-of-the-art results [29]. Despite their advantages, transformer-based models have certain limitations. One major drawback is their high computational demand; due to their size and complexity, these models require substantial computational resources and training time [74].

4.1.4 Hybrid-Based Methods

Hybrid models combine different classes of DDI extraction methods—rule-based, ML, and DL—to improve overall performance. These approaches aim to leverage the strengths of each method while mitigating their limitations. For example, a combination of rule-based methods and transformer models was explored in [57]. Additionally, ML was integrated with DL in [27].

4.2 Comparison of DDI Extraction Methods

As discussed before, various approaches have been proposed for DDI extraction, an essential task in pharmacovigilance and biomedical text mining. Each approach has its methodologies, advantages, and limitations. In Table 1.4, we provide a comparative analysis based on three primary methods: rule-based, ML-based, and DL-based approaches.

Method	Description	Advantages	Limitations
Rule-based	Uses predefined linguistic and syntactic rules to extract patterns from text.	Simple, can achieve high accuracy	Heavily relies on domain experts to develop extensive rule sets
ML-based	Employs supervised ML models such as SVM and logistic regression, trained on labeled data.	Do not require handcrafted rules.	Heavily depends on manual feature engineering
CNN-based	uses CNNs to automatically learn feature representations from raw text.	Captures local dependencies and hierarchical patterns effectively.	Struggles with sequential data and long-range dependencies.
RNN-based	uses RNNs to process sequences by maintaining memory of previous inputs.	Retains contextual information over time	Struggles with very long-range dependencies, slow training
Transformers-based	utilizes self-attention mechanisms to process entire sequences in parallel, capturing long-range dependencies.	Dynamically weighs the importance of different words in the input, effectively modeling complex contextual relationships.	Requires large datasets, has high computational cost, and is less interpretable.

Table 1.4: Comparison of various methods for text processing

5 Conclusion

In this chapter, we provided a comprehensive review of DDI extraction. We introduced the concept of drug-drug interactions, discussed their clinical significance, and explored the various data sources available. We also examined the role of NLP in biomedical text analysis and presented state-of-the-art methods for DDI extraction. We found that methods using transformers showed superior results. Although transformers still have limitations, they remain the best approach.

In the next chapter, we will present the fundamental concepts necessary for our project.

CHAPTER 2

THEORETICAL BACKGROUND

1 Introduction

This chapter will present an overview of the theoretical foundations required for our project. We will start by explaining NLP and its path with DL. Next, we will present the datasets and benchmarks available for extracting DDI. Finally, we will discuss the evaluation methods for these approaches.

The background information presented in this chapter will enhance our understanding of the methodologies and models employed in our study.

2 Fundamentals of NLP

NLP is the area of AI that facilitates the ability of machines to correctly interpret, understand, and produce human language through its written, spoken, and structured formats. NLP grows out of computational linguistics, the application of various principles from computer science to study linguistic patterns and structures [71].

2.1 NLP Concepts

In this section, we introduce some fundamental NLP concepts related to text preprocessing.

2.1.1 Tokenization

Tokenization is one of the first steps in any NLP pipeline. It is the process of breaking unstructured data and natural language text into smaller pieces called tokens so they can be analyzed separately [71]. Some tokenization methods are presented in Table 2.1.

Type	Definition
Word Tokenization	Splitting a sentence into individual words.
Sentence Tokenization	Breaking a paragraph into separate sentences.
Regular Expression Tokenization	Using patterns to split text based on specific rules.

Table 2.1: Types of tokenization

Text data can be tokenized through various approaches. One fundamental technique involves using regular expressions (regex), which also enables conditional tokenization.

2.1.2 Part-Of-Speech Taging

Part-of-speech (POS) tagging involves assigning labels to words in a sentence based on their grammatical categories. This assignment can depend on a word's definition as well as its contextual relationship with surrounding words in a phrase, sentence, or paragraph [71]. POS tags identify the grammatical function of each word (noun, verb, adjective, etc.), helping to determine its role within the sentence. Without POS tagging, words with multiple meanings could be misinterpreted, leading to errors in NLP applications. Consider these two examples in English:

- *"I like apples."* (Here, *"like"* works as a **verb** meaning "to enjoy.")
- *"This book is like a dictionary."* (Here, *"like"* works as a **preposition** meaning "quite similar to.")

POS tagging resolves such ambiguities by distinguishing between different grammatical roles of the same word.
A detailed description of POS tag categories is provided in Appendix A.

2.1.3 Dependency Parsing

Dependency parsing is a key technique for analyzing sentence structure. It constructs a dependency tree that captures the grammatical hierarchy of a sentence, with nodes representing words and directed edges indicating syntactic relationships, such as which word governs or modifies another [75].
There are two main techniques used for dependency parsing [75]:

- **Transition-Based Parsing** : Transition-based parsing is a machine learning-based approach to dependency parsing. It involves predicting a sequence of actions to build a dependency tree. These actions include shifting a word onto the stack, reducing the stack by creating a dependency between the top two words, or creating a new dependency by combining two existing dependencies.
- **Graph-Based Parsing** Graph-based parsing involves building a graph of the sentence where nodes represent words, and edges represent the dependencies between them. The graph is then analyzed using algorithms to identify the most likely dependency tree.

In Figure 2.1, we illustrate an example of dependency parsing

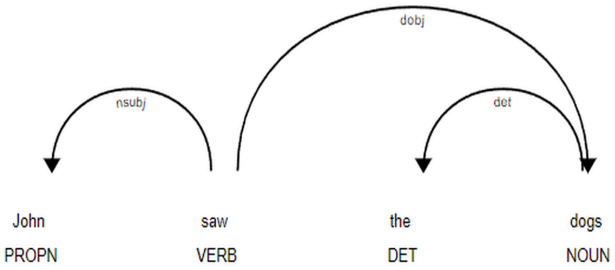


Figure 2.1: Example of dependency parsing

2.2 Word Representation

Strings and plain text cannot be processed directly in their raw form by ML and DL algorithms; they must be converted into numerical values to serve as inputs for any tasks. This transformation is broadly categorized into two approaches: **word encoding** and **word embeddings** (contextual representations).

2.2.1 Word Encoding

Word encoding techniques transform words into numerical formats without capturing semantic meaning. Common methods include:

- **Label encoding**: Label encoding is simply giving each unique category a unique integer, effectively converting categorical data into numerical values. This type of encoding is suitable for categorical data where there is an inherent order or ranking among the categories.
- **One-hot-encoding**: Unlike label encoding, which can introduce unintended ordinal relationships, one-hot encoding eliminates this issue by representing each category as a unique binary vector. Each categorical value is converted into a new column, and a binary value of 1 or 0 is assigned to indicate the presence or absence of the category [8]. This ensures that all categories are treated equally without implying any ranking or order. Table 2.2 illustrates an example of one-hot encoding for color categories.

Color	Red	Green	Blue
Red	1	0	0
Green	0	1	0
Blue	0	0	1

Table 2.2: One-Hot Encoding Example for Color Categories

While one-hot encoding prevents ordinal misinterpretations and works well for small categorical sets, it has drawbacks. High dimensionality increases memory. The

resulting sparse representation with lots of zeros and the failure to capture relationships between categories, treating them as entirely independent, can negatively affect some models [8].

Additional word representation methods, such as Term Frequency–Inverse Document Frequency (TF-IDF) and Bag-of-Words (BoW), are discussed in Appendix A.

2.2.2 Word Embeddings

Word embeddings are mathematical representations of words or phrases as vectors of real numbers in a high-dimensional space, where the position of each word encodes its semantic meaning and relationships [71].

- **Word2Vec**: Word2Vec is a technique that converts words into vector representations, capturing their meaning, semantic similarity, and relationships with surrounding text [76]. The fundamental idea behind Word2Vec is to represent each word as a multi-dimensional vector, where its position in this high-dimensional space encodes its semantic meaning, as illustrated in Figure 2.2.

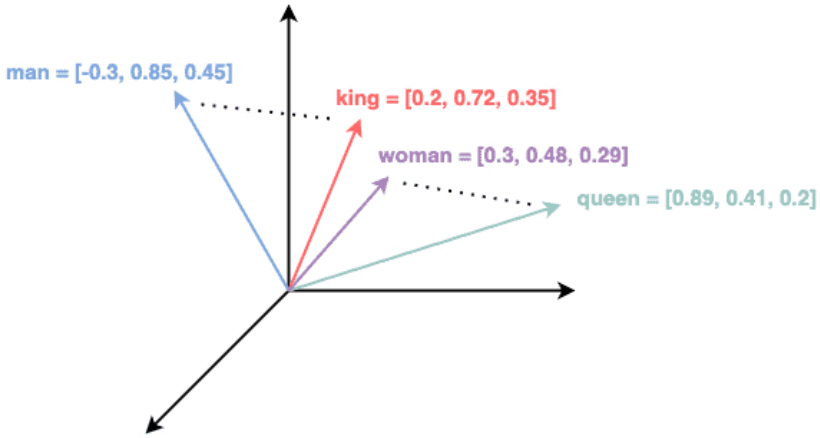


Figure 2.2: Word2Vec embeddings

Word2Vec has two primary neural network (NN) based variants [71]:

  - **Skip-Gram**: This model predicts context words (surrounding words) given a specific target word. It is particularly effective for large datasets, as it captures meaningful representations even for rare words.
  - **Continuous Bag of Words** : Unlike Skip-Gram, Continuous Bag of Words (CBOW) predicts a target word based on its surrounding context. It is computationally more efficient and faster, making it well-suited for smaller datasets.

- **Global Vector for Word Representation** : Global Vector for Word Representation (GloVe) is a word embedding method developed at Stanford that captures global relationships among words using a co-occurrence matrix. Unlike Word2Vec, which learns word relationships based only on local context, GloVe combines both local context (like the skip-gram algorithm) and global matrix factorization techniques, such as Latent Semantic Analysis (LSA). However, while LSA leverages

statistical information, it results in a suboptimal vector space structure and does not perform well on word analogy tasks. GloVe overcomes these limitations by incorporating global word co-occurrence statistics, leading to a more meaningful word representation. It determines how often words appear together within a fixed window size and uses this information to capture semantic relationships. Another key factor is the probability ratio of word occurrences. For example, words related to "ice" (e.g., "solid") will have high co-occurrence ratios, while unrelated words (e.g., "clothing") will have a ratio close to one. This structured approach to word co-occurrence makes GloVe more effective in distinguishing relevant from irrelevant words compared to raw probability-based models [76].

- **Transformer-based models**: Models based on the Transformer architecture, such as BERT, have revolutionized word representations by capturing contextual meanings more effectively than traditional methods like Word2Vec or GloVe. Unlike Word2Vec, which creates static embeddings for words, BERT generates dynamic, context-aware representations by analyzing entire sentences bidirectionally. This allows BERT to handle the presence of multiple meanings of a particular word in different contexts much more effectively. Additionally, transformers rely on self-attention mechanisms, enabling them to capture long-range dependencies in the text more efficiently than previous approaches [9]. Further details on Transformers and BERT will be discussed later in this chapter.

3 DL for NLP

DL has completely transformed NLP by switching the methodology from a rule-based approach to a flexible data-driven approach that can, depending on the corpus and learn complex patterns. Models such as Transformers and Graph Neural Networks (GNNs) are essential in capturing relationships and context present within text to improve performance, enhancing performance across various tasks.

3.1 Transformers

Transformers are NNs designed for sequence-to-sequence tasks. Introduced in the 2017 Google paper [67]. Unlike LSTMs, which process data sequentially and train slowly, Transformers leverage attention mechanisms within an encoder-decoder architecture, eliminating recurrent and convolutional layers. This enables parallel processing, faster training, and enhanced performance in NLP tasks.

3.1.1 Key Concepts

Self-Attention
Self-attention allows the model to focus on important words in a sentence by assigning different importance scores to different words. Unlike traditional models that process each word independently, self-attention allows words to influence each other dynamically. effectively modelling contextual relationships [65].It uses query (q), key (k), and value (v) vectors for each word, computed as follows [65]:

- Query vector q: Represents the word currently being processed and is used to find relevant words in the sequence.

- Key vector k: Represents all words in the sequence and is used to determine how much attention each word should receive.
- Value vector v: Contains the actual word representations that will be weighted based on the attention scores.

Each word's embedding is multiplied by three trained weight matrices to generate these vectors:

$$q(i) = W_q x(i) \quad \text{for } i \in [1, T] \tag{2.1}$$

$$k(i) = W_k x(i) \quad \text{for } i \in [1, T] \tag{2.2}$$

$$v(i) = W_v x(i) \quad \text{for } i \in [1, T] \tag{2.3}$$

The index i refers to the token index position in the input sequence, which has length T.

The self-attention score between words is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.4}$$

Where:

- d_k is the dimension of the key vectors.
- The softmax function ensures that the attention scores sum to 1.
- The division by $\sqrt{d_k}$ prevents extremely large values that could slow down learning.
- both q(i) and k(i) are vectors of dimension dk. The projection matrices Wq and Wk have a shape of dk×d, while Wv has the shape dv×d, where d represents the size of each word vector.

Multi-Head Attention

Instead of relying on a single set of attention scores, the Transformer uses Multi-Head Attention, which applies multiple self-attention mechanisms in parallel. Each attention head learns different aspects of the relationships between words. Each head has its query, key, and value matrices, and their outputs are concatenated and projected into a final representation [65]:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \text{head}_2, ..., \text{head}_h)W^O \tag{2.5}$$

where W^O is a learned projection matrix that combines the outputs of different heads. Figure 2.3 summarizes the multi-head attention mechanism.

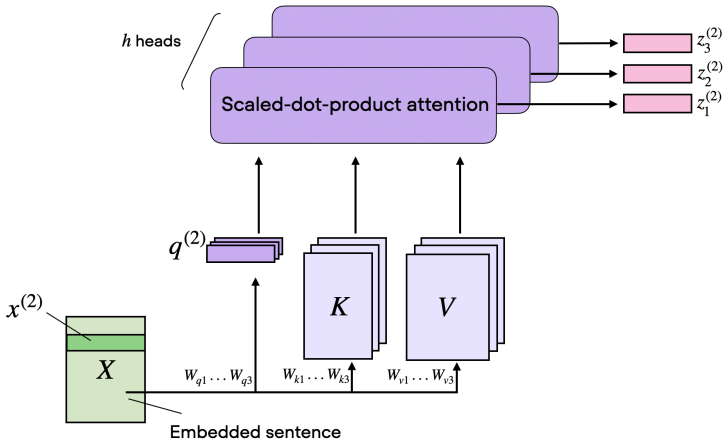


Figure 2.3: Multi-head self-attention mechanism

Positional Encoding

As the Transformer processes an entire sequence in parallel, each word in a sentence simultaneously flows through the Transformer's encoder/decoder stack. The model itself does not inherently encode positional information. Understanding the relative positions of words is crucial for capturing meaning in languages. To address this, positional encoding is added to the Transformer architecture, allowing the model to incorporate order information [61].

One approach to providing the model with positional awareness is to add a position-dependent vector to each word embedding. Instead of learning positions through training, the Transformer employs a fixed d-dimensional encoding based on sine and cosine functions. Given a token position t, its encoding $\mathbf{p_t} \in \mathbb{R}^d$ is defined as:

$$p_t(i) = \begin{cases} \sin(\omega_k \cdot t), & \text{if } i = 2k \\ \cos(\omega_k \cdot t), & \text{if } i = 2k+1 \end{cases} \tag{2.6}$$

where

$$\omega_k = \frac{1}{10000^{\frac{2k}{d}}} \tag{2.7}$$

This formulation ensures that the encoding varies smoothly across dimensions, creating a structured representation of positions. The frequencies decrease along the vector dimension, forming a geometric progression from 2π to $10000 \cdot 2\pi$ on the wavelengths.

The positional encoding is then summed element-wise with the word embeddings before being fed into the model:

$$\mathbf{e'_t} = \mathbf{e_t} + \mathbf{p_t} \tag{2.8}$$

where $\mathbf{e_t}$ represents the embedding of token t. Ensuring that both embeddings and positional encodings have the same dimensionality allows this summation to be performed directly. This addition provides the model with position-dependent information without introducing additional trainable parameters [61].

3.1.2 Encoder-Decoder Structure

The Transformer follows an **encoder-decoder** structure, where **positional encoding** is applied to both the input and output embeddings before they are processed [5]. Figure 2.4 illustrates this structure.

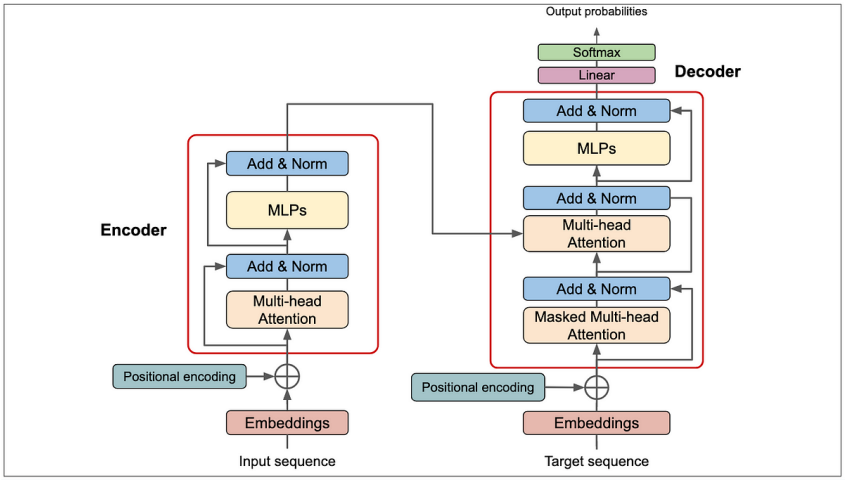


Figure 2.4: The architecture of the Transformer

The **encoder** consists of N identical layers, each containing two sub-layers [5]:

- A multi-head self-attention mechanism, which allows the encoder to attend to all positions in the input sequence simultaneously.
- A position-wise fully connected feedforward network, which consists of two linear transformations with a ReLU activation in between.

Similarly, the **decoder** consists of N identical layers, with an additional encoder-decoder attention mechanism as a third sub-layer.

- The encoder-decoder attention allows the decoder to focus on relevant parts of the input sequence.
- The decoder's self-attention is masked, preventing it from attending to future positions in the output sequence.
- Finally, the output of the last decoder layer passes through a linear layer followed by a softmax function, mapping it to a probability distribution over the target vocabulary.

3.2 Pre-trained Large Language Models for Feature Extraction

Training large transformer models from scratch requires enormous computational power and big domain-specific datasets [62]. To avoid this issue, pre-trained Transformer-based Large Language Models (LLMs) are constructed with general and domain-specific corpora, allowing researchers to fine-tune them for specific biomedical NLP tasks such as NER, RE, and document classification [60]. These models are typically categorised based on architecture, encoder-only and decoder-only [63]:

- **Encoder-only models**: such as Bidirectional Encoder Representations from Transformers (BERT), which is pre-trained with masked language modeling (MLM) and leverages bidirectional context to acquire semantic dependencies in both directions. Such bidirectionality is helpful for feature extraction in biomedical NLP, where token context understanding is crucial.

- **Decoder-only models**: such as Generative Pre-trained Transformer (GPT) [47], are trained in a unidirectional context (from left-to-right) and are useful for text generation tasks.

3.2.1 BERT

BERT, introduced by Google researchers [17], was a turning point model in NLP due to its successful application of unsupervised pretraining. It was pre-trained on massive amounts of unlabelled, plain datasets, including the entire English Wikipedia and Brown Corpus. To accommodate different computational requirements and application needs, BERT is available in multiple model sizes, varying in the number of parameters, layers, and memory footprint, as shown in Table 2.3:

Model	Parameters (Millions)	Layers	Memory (MB)
BERT-Base	110	12	420
BERT-Large	340	24	1,600
BERT-Mini	11	4	30
BERT-Tiny	4.4	2	6

Table 2.3: Different sizes of BERT models.

Pre-training Objectives
Bert uses two new training strategies, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP):

- **Masked Language Model** consists of giving BERT a sentence and optimizing the weights inside BERT to output the same sentence on the other side. So we input a sentence and ask that BERT outputs the same sentence, some words are masked (replaced with [MASK]), and BERT learns to predict those words from their context. This helps BERT grasp how words relate to each other, both before and after MLM keeps the word in focus from seeing itself or having a fixed meaning independent of its context. In BERT, words are defined by their surroundings, not by a prefixed identity.

- **NSP** enables BERT to learn the relationships between words within a sentence and the relationships between sentences in a document or text corpus. Here's an explanation of how NSP operates in BERT:

  1. **Sentence Pair Input:** BERT takes a pair of sentences as input during pre-training. These sentences are typically sampled from a large text corpus. In the input, there is a special separator token (e.g., [SEP]) between the two sentences.
  2. **Binary Classification:** The NSP task is a binary classification task. BERT learns to predict whether the second sentence in the pair logically follows the first sentence (i.e., is the "next sentence") or not. This task helps the model capture the relationships and cohesiveness between sentences in a document.
  3. **Training Objective:** During training, BERT is presented with pairs of sentences, and it learns to predict whether they are connected or not. The model is trained to produce two possible labels:

- **Label 1 (IsNext):** If the second sentence logically follows the first sentence, the label is set to "IsNext".
- **Label 2 (NotNext):** If the second sentence is not the next, the label is set to "NotNext".

4. **Loss Function:** BERT is trained to minimize a loss function, such as cross-entropy loss, over these binary classification labels. The goal is to make accurate predictions about whether the second sentence is a continuation of the first or not.

By pre-training on a large text corpus with both the NSP task and MLM, BERT learns to generate contextual embeddings of words and sentences, capturing contextual nuances in a bidirectional manner. These pre-training tasks enable BERT to generate rich contextualized representations, which are utilized by its feature extraction mechanisms. Specifically, BERT's architecture employs token, segment, and position embeddings, along with the special [CLS] token, to produce embeddings that encode both word-level and sentence-level information for downstream tasks.

Feature Extraction in BERT

Tokenization is the first critical step in BERT feature extraction. BERT does not use simple word-level tokenisation but makes use of WordPiece tokenisation, which breaks words into subword units [13]. For example, a word like "eating" might be tokenised into "eat" and "##ing", allowing BERT to generalise across rare or unseen words. Special tokens are added by BERT to format its input sequence [32]:

- **[CLS]**: Positioned as the beginning of the input sequence, this token surrounds the meaning of the entire sequence. Its final hidden state is used in sentence classification tasks on a sentence level.
- **[SEP]**: Used to separate pairs of sentences or to mark the end of a single sentence.
- **[PAD]**: Pads shorter sentences to a fixed maximum length with the same-size inputs. An attention mask separates [PAD] tokens from meaningful tokens so that the model will not attend to padding during processing.

To preserve token sequential order, BERT uses position embeddings, where each token's position within the input sequence is represented [37]. Since transformers lack inherent knowledge of word order, these embeddings provide critical syntactic information, allowing BERT to capture the relative positions of words and their contextual relationships. Each token's position is represented as a learnt vector, combined with its token and segment embeddings to form the final representation of the input.
Segment embeddings also enhance BERT's ability to process more than a single sentence. They assign different vectors to different sentences such that the model learns to differentiate between contexts in the input sequence. All together, token, position, and segment embeddings create contextualised representations that capture word-level and sentence-level information, and thus BERT is extremely flexible for downstream NLP tasks.

3.2.2 Domain-Specific Variants of BERT

While BERT was trained on general-domain corpora such as Wikipedia and BookCorpus, researchers have developed domain-specific variants by pretraining the same architecture on specialized datasets. These models are optimized for tasks within particular fields, allowing for better performance in specific fields without the need for full-scale retraining.

- **BioBERT** [33]: trained on large-scale biomedical corpora. It was trained on 4.5 billion words from PubMed abstracts and 13.5 billion from PubMed Central full-text articles. Compared to previous biomedical text mining models, which primarily focus on a single task, such as NER or QA, BioBERT improves performance on a range of biomedical NLP tasks. , including NER, RE, and QA.

- **SciBERT** [12]: developed for scientific text processing. It was trained on a corpus of 1.14 million papers from the Semantic Scholar Open Research Corpus, comprising 18% computer science and 82% biomedical papers. It uses a domain-specific scientific vocabulary and was trained from scratch, leading to improved performance on scientific NLP tasks such as named entity recognition and text classification.

- **PubMedBERT** [24]: trained from scratch using only biomedical text from PubMed. The pretraining corpus consists of 14 million PubMed abstracts totaling 3.1 billion words. PubMedBERT is fully trained within the biomedical domain. This enables it to better capture medical terminology and achieve superior performance across a range of biomedical NLP tasks.

3.3 Fine-tuning for LLMS

Transfer learning is a machine learning technique in which knowledge gained through one task or dataset is used to improve model performance on another related task and/or a different dataset. Transfer learning uses what has been learnt in one setting to improve generalisation in another setting [59].
Fine-tuning is a form of transfer learning, where the knowledge learned from a large-scale, general training process is transferred to a more specific task. In the context of LLMs, it involves taking a pre-trained general-purpose model and continuing its training on a smaller, domain-specific dataset. This process enables the model to adapt to specialized tasks or domains while retaining its general language capabilities. [22]. Recent research has shown that fine-tuning pre-trained LLMs can achieve or surpass state-of-the-art performance on many tasks, especially when combined with domain-specific data [44, 10].

3.4 Graph Neural Networks

3.4.1 Graph Structures

Graphs are data structures used to model irregular and non-linear relationships. A graph consists of a set of nodes and a set of edges that join the nodes. There are 2 types of graphs [28]:

- Directed: In a directed graph, edges have a direction indicating the flow or relationship between nodes.

- Undirected: In an undirected graph, edges have no direction and represent symmetric relationships.

In the context of NLP, we can consider the text as a graph, using nodes to represent words or sentences and edges to represent syntactic or semantic relations.
A common way to represent a graph G is through its adjacency matrix A, where $A_{i,j} = 1$ if there is an edge between node v_i and node v_j, and $A_{i,j} = 0$ otherwise [28].

3.4.2 Overview of GNNs

GNNs are types of NN that are designed to handle structured graph data. They can be applied to a variety of tasks [73], including:

- **Node Level task:** These involve taking information from a node's local neighborhood to predict the class of individual nodes based on the graph's structure or node attributes, or to predict relationships between different nodes.
- **Graph Level task:** These involve classifying entire graphs based on their overall structure or node properties.

In a GNN, each node maintains a feature vector and updates iteratively by aggregating information from its neighbours. Learn embeddings for each node in the graph and aggregate the embeddings of the nodes to produce the embeddings of the graph [3].

Generally, the learning process of node embeddings utilises graph structure and input node embeddings, which can be summarised by the following operation:

$$\mathbf{h}_i^{(l)} = f_{\text{filter}}(A, \mathbf{H}^{(l-1)}) \tag{2.9}$$

Where:

- $\mathbf{h}_i^{(l)}$: Feature vector of node i at layer l.
- f_{filter}: The message-passing or filtering function.
- A: Adjacency matrix.
- $\mathbf{H}^{(l-1)}$: Matrix of all node features from the previous layer.

In this operation, the filter function f_{filter} takes the graph structure (A) and the previous layer's node features ($\mathbf{H}^{(l-1)}$), and computes a new feature vector for each node by aggregating information from its neighbors. This process, known as graph filtering, enables each node to update its representation based on the features of its local neighborhood [3].

Graph Filtering
Graph filtering refers to the process of aggregating and transforming node features based on the graph's structure. There are several common approaches to designing the graph filter function f_{filter}. The most used include [3] :

- **Spectral-based filters**: These rely on spectral graph theory and typically involve operations in the frequency domain of the graph.

- **Spatial-based filters**: These directly aggregate information from a node's local neighborhood in the graph structure.
- **Attention-based filters**: Inspired by the self-attention mechanism, these assign varying weights to neighboring nodes during message passing

Two popular GNNs that implement distinct graph filtering functions f_{filter} are described below:

- **Graph Convolutional Network (GCN)** [30]: GCNs use spectral-based graph filters inspired by graph signal processing, generalizing CNNs to graphs with linear computational complexity and constant learning complexity. The filtering operation for a node's feature vector at layer l is defined as:

$$\mathbf{H}^{(l)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\mathbf{H}^{(l-1)}\mathbf{W}^{(l-1)}\right) \tag{2.10}$$

  where $\tilde{A} = A + I_n$ is the adjacency matrix with self-loops, $\tilde{D}$ is the degree matrix of $\tilde{A}$, $\mathbf{H}^{(l-1)} \in \mathbb{R}^{n\times d}$ is the node feature matrix at layer $l-1$, $\mathbf{W}^{(l-1)}$ is a trainable weight matrix, and $\sigma(\cdot)$ is an activation function.

- **Graph Attention Network (GAT)** [68]: GATs employ attention-based filters, using multi-head attention to dynamically assign importance weights to neighboring nodes during message passing. For the final layer (i.e., the L-th layer), the node embeddings are updated by averaging K attention heads:

$$\mathbf{h}_i^{(L)} = f_{\text{filter}}(A, \mathbf{H}^{(L-1)}) = \sigma\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{v_j\in\mathcal{N}(v_i)}\alpha_{ij}^k\vec{W}_k^{(L)}\mathbf{h}_j^{(L-1)}\right) \tag{2.11}$$

  where α_{ij}^k are attention coefficients for head k, $\vec{W}_k^{(L)}$ is the weight matrix for head k, $\mathcal{N}(v_i)$ is the 1-hop neighborhood of node v_i (including itself), and $\sigma(\cdot)$ is an activation function.

Message Passing

The key challenge in modeling graph-structured data is effectively capturing the dependencies among nodes. Unlike images or sequences, graphs lack a regular grid structure or linear order. Message passing overcomes this by letting each node share information with its neighbors and then update its representation based on the aggregated messages [73].

At each iteration of the message passing in a GNN, the hidden state of node u, denoted $\mathbf{h}_u^{(k)}$, is updated using information from its neighbors $\mathcal{N}(u)$:

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\big(\mathbf{h}_u^{(k)}, \text{AGGREGATE}^{(k)}\{\mathbf{h}_v^{(k)} : v \in \mathcal{N}(u)\}\big). \tag{2.12}$$

Here:

- $\mathbf{h}_u^{(0)} = \mathbf{x}_u$ is the initial feature vector of node u.
- $\text{AGGREGATE}^{(k)}$ collects and combines messages from neighbors (e.g., by summation or averaging).

- UPDATE$^{(k)}$ merges the aggregated message with the node's previous state (often via a neural network layer).

After K such iterations, the final embedding $\mathbf{z}_u = \mathbf{h}_u^{(K)}$ encodes both the node's features and its multi-hop context. Because aggregation treats neighbour messages as an unordered set, GNNs are naturally permutation equivariant—insensitive to the order in which neighbours are processed.

GNNs require an initial feature vector $\mathbf{x}_u$ for each node $u \in \mathcal{V}$ to initiate the message passing process [28].

4 Evaluation of Automatic DDI Extraction Models

DDI extraction represents a supervised learning challenge that demands a methodical evaluation approach. The assessment framework typically consists of two distinct phases:

- **Training phase:** where the model learns patterns or extracts features from labelled data.
- **Testing phase:** where the trained model is evaluated and tested on unseen data using specific performance metrics.

Consequently, practical evaluation requires high-quality datasets and benchmarks that are divided into separate training and testing sets. This division is crucial for accurately assessing the model's ability to generalise its learning to new, unseen examples.

4.1 Evaluation Metrics

One of the most widely used evaluation metrics is the F1-score, which balances precision and recall to provide a single measure of a model's performance. F1-score is defined as :

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.13}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.14}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.15}$$

where:

- TP (True Positives): The number of instances correctly classified as positive
- FP (False Positives): The number of instances incorrectly classified as positive
- FN (False Negatives): The number of instances incorrectly classified as negative
- TN (True Negatives): The number of instances correctly classified as negative

When the F1-score is applied to multiclass classification problems, three common variants are used:

- **Micro-averaged**: Calculates precision and recall globally by aggregating the total TP, FP, and FN across all classes before computing the F1-score.

- **Macro-averaged F1-score** : Computes F1-score for each class separately and then averages them equally.

which are defined as follows :

$$\text{Precision}_{\text{micro}} = \frac{\sum TP}{\sum TP + \sum FP} \tag{2.16}$$

$$\text{Recall}_{\text{micro}} = \frac{\sum TP}{\sum TP + \sum FN} \tag{2.17}$$

$$F1_{\text{micro}} = 2 \times \frac{\text{Precision}_{\text{micro}} \times \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}} \tag{2.18}$$

$$F1_{\text{macro}} = \frac{1}{N} \sum_{i=1}^{N} F1_i \tag{2.19}$$

where $F1_i$ is the F1-score of class i
In the case of the DDI extraction task, the primary evaluation metric widely used is F1-macro.

4.2 Benchmarks and Available Datasets

Many datasets are available for DDI extraction, but only a few are considered gold standard. These datasets are manually annotated by experts and widely accepted in the research and healthcare communities. The most well-known gold standard datasets for DDI extraction are DDI Extraction 2013, TAC 2018, and TAC 2019, which are commonly used for training and evaluation.

4.2.1 DDI Extraction 2013

The DDI Extraction 2013 corpus [26] is a manually annotated DDI corpus based on DrugBank and MedLine abstracts and is the main corpus to compare different DDI extraction methods. It contains five predefined classes:

- **Mechanism**: Refers to a pharmacokinetic mechanism description.
  For example: "Withdrawal of rifampin decreased the warfarin requirement by 50%."

- **Effect**: Refers to the effect or the pharmacodynamic mechanism of interaction.
  For example: "PGF2alpha produced significantly increased vasoconstriction after a single administration of oxytocin."

- **Advice**: Refers to a recommendation or advice regarding a DDI.
  For example: "Caution should be exercised when alosetron and ketoconazole are administered concomitantly."

- **Int**: Refers to an interaction between drugs without providing more information.
  For example: "Clinical implications of warfarin interactions with five sedatives."

The detailed statistics of each class corpus are provided in Table 2.4

	Documents	Sentences	Drug pairs	Negative pairs	Positive pairs			
					Adv.	Eff.	Int.	Mec.
Train	714	5805	26022	22231	826	1687	189	1319
Test	191	1299	5716	4737	221	360	96	302

Table 2.4: Statistics of DDI Extraction 2013 Dataset

4.2.2 TAC 2018 and TAC 2019 DDI Extraction

The TAC 2018 DDIs track dataset [20] includes 325 structured product labels (SPLs), which are divided into a training set with 22 drug labels and a test set with 57 drug labels. The type of DDIs is divided into three groups:

- **Pharmacokinetic (PK)**: Refers to a pharmacokinetic mechanism description.
- **Pharmacodynamic (PD)**: Refers to the pharmacodynamic mechanism of interaction.
- **Unspecified (U)**: Refers to an interaction between drugs without providing more information.

The TAC 2019 DDIs Extraction dataset [56] contains 406 SPLs, split into a training set of 211 drug labels and a test set of 81 drug labels. The DDI types are consistent with those in TAC 2018. The primary difference between TAC 2019 and TAC 2018 DDIs Extraction is that while TAC 2018 included information from SPLs as well as other text types such as literature and social media, TAC 2019 only uses SPLs. The primary difference between TAC 2018 and TAC 2019 is that TAC 2018 includes additional sources such as biomedical literature and social media, while TAC 2019 exclusively uses SPLs. Detailed statistics for both corpora are presented in Table 2.5.

DDIs	**Train**		**Test**	
	TAC 2018	**TAC 2019**	**TAC 2018**	**TAC 2019**
Pharmacodynamic (PD)	47	553	335	292
Pharmacokinetic (PK)	60	494	296	118
Unspecified	62	665	440	202

Table 2.5: Statistics for TAC 2018 and TAC 2019 datasets

5 Conclusion

In this chapter, we outlined the fundamental concepts related to our work. Additionally, we reviewed the available datasets and benchmarks and explored how to evaluate an automatic DDI extraction model.

In the next chapter, we will detail our proposed approach, which includes data preprocessing steps, model architectures, and techniques employed.

CHAPTER 3

PROJECT DESIGN AND METHODOLOGY

1 Introduction

This chapter outlines the project design. It starts by introducing the chosen dataset and describing its structure. Then, it provides an overview of the proposed solution. Finally, it details the system architecture, data preprocessing steps, and specific techniques used in the design process.

2 Data Selection

As explained in the earlier chapter, the benchmarks and datasets commonly used for DDI extraction studies are DDI Extraction 2013, TAC 2018, and TAC 2019. In our case, we chose DDI2013 because it provides high-quality annotations tailored for DDI extraction tasks, including both Drug-Named Entity Recognition (DNER) and relation extraction. Researchers widely use it due to its comprehensive documentation and the extensive research and articles built upon it, which facilitates benchmarking and comparison.

2.1 Data Description

DDI Extraction 2013 includes annotated sentences from MEDLINE abstracts and the DrugBank database. MEDLINE contains biomedical research articles, while DrugBank consists of manually curated texts collected from various sources and verified by experts. The dataset provides the training and test instances as sentences. If a sentence contains more than two drug names, all possible drug pairs in the sentence are annotated separately. This means that a single sentence can have many instances.

2.2 Data Structure

The dataset directory is split into two subdirectories, DRUBANKS and Medline :

- DrugBank contains 572 documents describing DDI from the DrugBank database
- Medline contains 142 documents describing DDI from the Medline database

The documents are XML files structured as shown in Figure 3.1

```
<?xml version="1.0" encoding="UTF-8"?>
<document id="DDI-DrugBank.d605">
    <sentence id="DDI-DrugBank.d605.s0" text="SKELAXIN may enhance the effects of alcohol, barbiturates and other
    CNS depressants.">
        <entity id="DDI-DrugBank.d605.s0.e0" charOffset="0-7"
            type="brand" text="SKELAXIN"/>
        <entity id="DDI-DrugBank.d605.s0.e1" charOffset="36-42"
            type="drug" text="alcohol"/>
        <entity id="DDI-DrugBank.d605.s0.e2" charOffset="45-56"
            type="group" text="barbiturates"/>
        <pair id="DDI-DrugBank.d605.s0.p0" e1="DDI-DrugBank.d605.s0.e0"
            e2="DDI-DrugBank.d605.s0.e1" ddi="true" type="effect"/>
        <pair id="DDI-DrugBank.d605.s0.p1" e1="DDI-DrugBank.d605.s0.e0"
            e2="DDI-DrugBank.d605.s0.e2" ddi="true" type="effect"/>
        <pair id="DDI-DrugBank.d605.s0.p3" e1="DDI-DrugBank.d605.s0.e1"
            e2="DDI-DrugBank.d605.s0.e2" ddi="false"/>
    </sentence>
</document>
```

Figure 3.1: DDI EXTRACTION 2013 DOCUMENT FORMAT

- <**document**>: The root element of any document with the following attributes:
  - `id`: unique id composed by:
    - The name of the corpus (DDI-DrugBank or DDI-MedLine)
    - An identifier beginning with "d" followed by a number
  - Each document contains many sentences
- <**sentence**>: Each sentence of the document is contained within a `<sentence>` element with attributes:
  - `id`: unique id starting with the id of the document (e.g., DDI-MEDLine.d127) and an id beginning with "s" followed by the sentence index in the current document (starting at 0)
    Example: `DDI-MEDLine.d127.s6` means the seventh sentence of document 127 of the MedLine corpus
  - `text`: Contains the content of the sentence

  Within the `<sentence>` element, there are:
  - `<entity>`: Annotated Drugs
  - `<pair>`: Annotated DDIs
- <**entity**>: Each `<entity>` element has the following attributes:
  - `id`: unique id starting with an id of the sentence (e.g., DDI-MEDLine.d127.s6), followed by an id beginning with "e", followed by the entity index (starting at 0)
    Example: DDI-MEDLine.d127.s6.e1
  - `charOffsets`: Start and end positions of the mention (separated by a dash)
  - `text`: Name of the mentioned entity (e.g., "digoxin")
  - `type`: Entity type (one of):

    - * `drug`: Human medicines known by generic name
    - * `brand`: Drugs described by trade or brand name
      (Note: A single generic drug can have multiple brand names.)
    - * `group`: Refers to a class or family of drugs sharing properties (e.g., NSAIDs, beta-blockers).
    - * `drug_n`: Refers to substances not approved for human use. (e.g, toxins, pesticides)

- <**pair**>: Each `<pair>` element has the following attributes:
  - `id`: unique id starting with an id of the sentence (e.g. DDI-MEDLine.d127.s6), followed by an id beginning with "p", followed by the pair index (starting at 0)
    Example: DDI-MEDLine.d127.s6.p1
  - `e1`: Id of the first interacting entity (e.g. DDI-MEDLine.d127.s6.e0)
  - `e2`: Id of the second interacting entity (e.g. DDI-MEDLine.d127.s6.e1)
  - `ddi`: Interaction flag:
    - * `true`: Interaction exists between e1 and e2
    - * `false`: No interaction
  - `type`: Type of DDI (if `ddi=true`):
    - * `advice`
    - * `effect`
    - * `mechanism`
    - * `int`

3 Overview of the Proposed Solution

In the current study, we explore the task of DDI extraction, which contains two closely interdependent subtasks: DNER and drug-drug interaction classification (relation extraction).
The objective of the DDI extraction task is to discover mentions of drug-named entities in biomedical text and extract the interaction relations between pairs of drug entities. While most previous works have focused only on the DDI extraction component with the assumption of pre-tagged drug names, our solution processes the two tasks simultaneously, handling the process from unannotated text to systematically arranged interactions.
Figure 3.2 illustrates an example of the DDI extraction task. The top part of the figure shows a biomedical sentence containing multiple drug mentions, including *phenothiazines*, *butyrophenones*, *risperidone*, *isoniazid*, and *levodopa*.
In the first step, DNER identifies these mentions in the sentence. Next, the interaction type among the drug pairs is determined by the drugs relation extraction model; namely, the drug combination (`cholestyramine`, `ezetimibe`) is of type `mechanism` interaction.

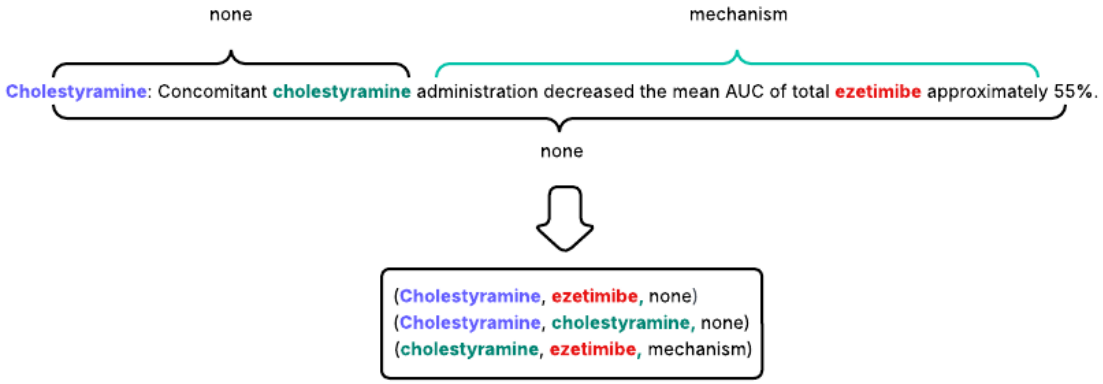


Figure 3.2: Example of a DDI extraction task

To handle the complexities involved in every subtask, we suggest a two-stage pipeline, one stage for each of the following subtasks:

- **DNER Model**: The first component is a variant of Bert fine-tuned for DNER, which is responsible for identifying drug mentions in the text and annotating them with appropriate entity types.
- **Drugs Relation Extraction Model** : The second component uses a hybrid architecture for DDI extraction.

Figure 3.3 illustrates the overall data flow from raw input to the final DDI prediction.

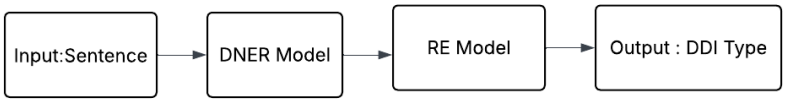


Figure 3.3: Overview of the proposed two-stage architecture for DDI extraction

In the sections that follow, we provide a detailed description of the two parts and the respective preprocessing steps associated with each.

4 Drug Name Entity Recognition Model

We use a BioBERT-based model for DNER, which identifies drug-related entities in text. The following subsections outline the preprocessing pipeline and model architecture employed for this task.

4.1 Preprocessing Pipeline

The DNER preprocessing pipeline, shown in Figure 3.4, begins by cleaning the input text and then applies BIO tagging to each word. Each step is described in detail in the following subsection.

Figure 3.4: DNER Preprocess Pipline

4.1.1 Data Cleaning

Data cleaning is typically the first step in data preprocessing. It involves removing empty instances and noise to improve data quality. In our case, we removed empty instances by eliminating <sentence> or < document> elements that are null. After this, to ensure consistency and simplify tokenization, all sentences are converted to lowercase.

4.1.2 BIO Tagging

To perform the classification task with BioBERT, we employed the BIO tagging scheme, a standard approach in NER tasks that distinguishes between the beginning (B) and the inside (I) of entities. O is used for non-entity tokens [42].
Table 3.1 illustrates the use of the BIO scheme for DNER in the sentence: *'not compatible with CNS depressants or tricyclic antidepressants'*.

Token	**BIO Label**
not	O
compatible	O
with	O
CNS	B-DRUG
depressants	I-DRUG
or	O
tricyclic	B-DRUG
antidepressants	I-DRUG

Table 3.1: Example of BIO Tagging for DNER

4.1.3 Tokenization and Label Alignment

Since the BioBERT tokenizer splits words into subword tokens, we align the BIO labels with these subword tokens to ensure accurate training. For example, a multi-word drug entity like "tricyclic antidepressants" may be tokenized into smaller units, requiring the `B-DRUG` and `I-DRUG` labels to be adjusted accordingly. This step ensures that the preprocessed data is compatible with BioBERT's tokenization.

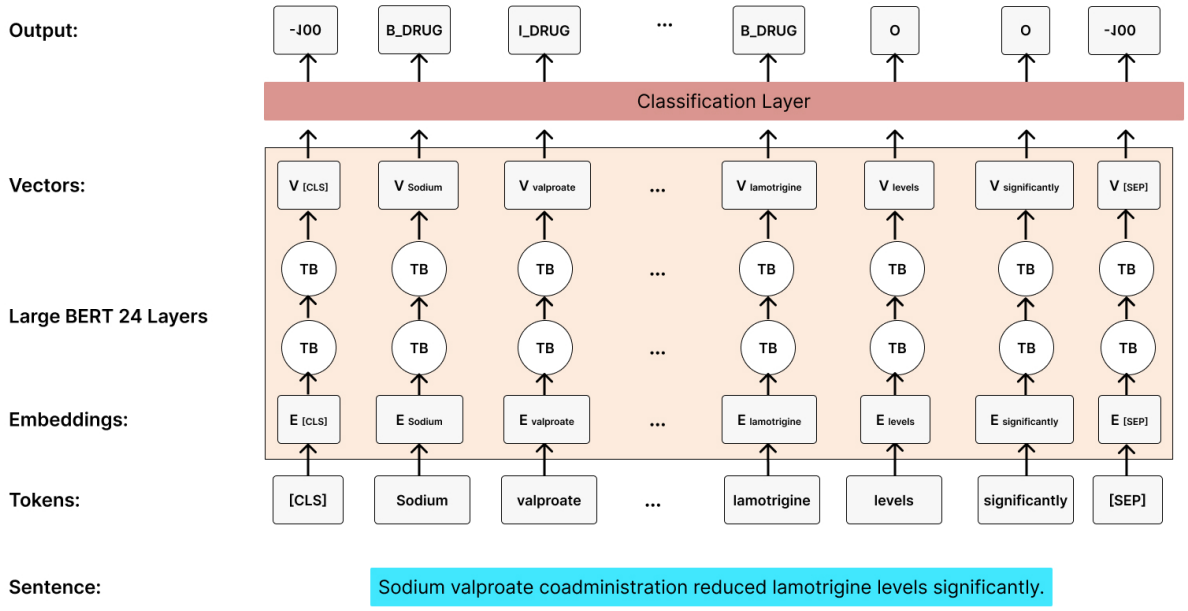
4.2 Model Architecture

We apply the **BioBERT** model to DNER due to its bias toward biomedical language understanding. Pre-trained on over a million PubMed abstracts and full-text papers, BioBERT has demonstrated outstanding performance on a variety of biomedical NER benchmark tasks, such as NCBI-disease and BC5CDR [33]. Its prowess is not only that it

can surpass general-purpose models like BERT, but also that it can receive fine-grained biomedical context, which is important to realize entity boundaries and semantics in domain-specific tasks like DNER.

Among the available variants, we selected **BioBERT-large** with 24 transformer layers and 340 million parameters, considerably more than BERT-base (12 layers, 110 million parameters), as reported in Table 2.3. The greater depth and representational ability enable it to identify more subtle semantic details common in biomedical text.
To further adapt the model to the characteristics of the DDI2013 dataset, we also fine-tuned BioBERT on this corpus. Fine-tuning task-specific pre-trained language models is a successful and general practice to promote downstream performance in biomedical NER.

As indicated in Figure 3.5, the input text is tokenized first with the WordPiece tokenizer from BioBERT. The obtained sequence is then input into the BioBERT encoder to get contextualized embeddings, which are subsequently given to a fully connected classification layer with Softmax activation to output BIO tags for each token. Special tokens, such as `[CLS]`, `[SEP]`, and `[PAD]` are ignored during training by assigning them a label of `-100` following practices in token-level classification tasks.



TB denotes the BERT Transformer Blocks.

Figure 3.5: DNER model architecture

5 Drugs Relation Extraction Model

To extract DDI types from biomedical text, we propose a hybrid model with a primary focus on BioBERT, motivated by its outstanding performance on biomedical relation extraction benchmarks, such as ChemProt and DDI [33], and its ability to produce rich semantic representations. Give a good semantic representation. While effective, models relying solely on BioBERT face several limitations. LLMs are susceptible to issues such as struggling to learn new data, susceptibility to hallucinations, lack of interpretability,

and overfitting on small datasets [72]. On the other side, models like BERT lack direct syntactic dependency encoding and can miss subtle inter-entity relationships [79], which are crucial to accurate DDI classification.

To address these issues, we suggest enhancing BioBERT with syntactic dependency information, which provides an additional representation that captures grammatical relations between entities. Dependency representations enable us to reveal how drugs are related to one another through the underlying linguistic structure of a sentence, which can potentially provide important relational information that is not included in the semantic embedding alone.
For example, consider the following sentence:

"The combined use of **fluconazole** with **cisapride** is contraindicated."

A dependency parser reveals that both "fluconazole" and "cisapride" appear in nested prepositional phrases modifying the noun "use." Meanwhile, the verb phrase "is contraindicated" assigns a clinical judgement to the entire compound subject. This structural analysis suggests that fluconazole and cisapride are being jointly referenced in a potentially adverse context, thereby supporting a DDI classification.

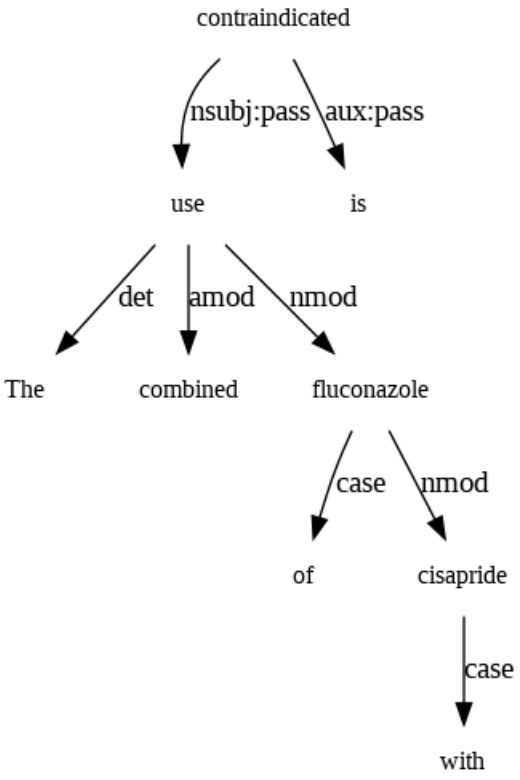


Figure 3.6: Dependency tree illustrating syntactic structure linking drug mentions.

The following subsections detail the preprocessing pipeline and the core components of our proposed model.

5.1 Preprocessing Pipeline

As illustrated in Figure 3.7, the preprocessing pipeline prepares the input text for DDI relation extraction by cleaning the data, handling class imbalance, and standardizing

drug mentions. Each step is explained in detail below.

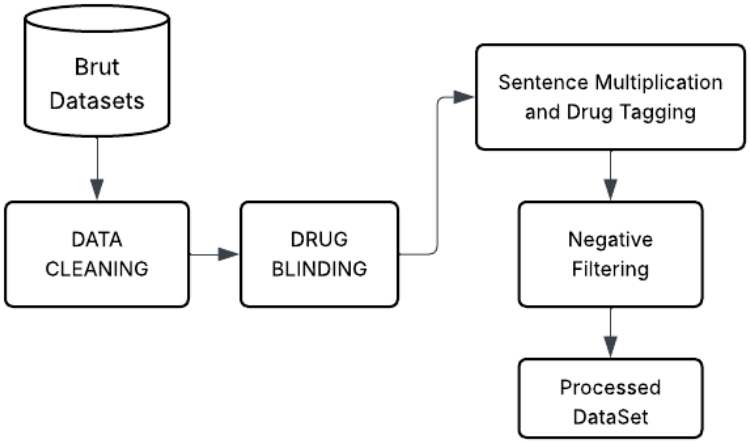


Figure 3.7: Drug RE Preprocess Pipeline

5.1.1 Data Cleaning

Like in our data cleaning process for DNER, we eliminated empty documents and sentences and converted the remaining sentences to lowercase. However, in this case, we also removed incomplete sentences containing fewer than two <entity> tags or lacking a <pair> tag. This is because such sentences do not provide sufficient information to identify drug interactions.

5.1.2 Drug Blinding

In this step, we replace all drug names in a sentence to `DRUG-{index}`, with the index representing the index of the drug in the sentence. Earlier studies [48, 21] reported that drug blinding helps in improving the model performance. While their approach typically replaces the two targeted drugs with `DRUG-A` and `DRUG-B`, and other drug names with `DRUG-N`, our method assigns a unique index to each drug to preserve referential identity, which is necessary for our downstream processing.
For example, consider the following sentence:

> "No clinically significant changes to **lamivudine** or **zidovudine** pharmacokinetics were observed following concomitant administration of **abacavir**."

The resulting output will be:

> "No clinically significant changes to **DRUG0** or **DRUG1** pharmacokinetics were observed following concomitant administration of **DRUG2**."

5.1.3 Sentence Multiplication and Drug Tagging

As explained in 2, if a sentence has multiple drug pairs, it will have many instances. To address this, we duplicate each sentence based on the number of drug pairs it contains. For instance, if a sentence has three pairs, we generate three corresponding copies. Each copy highlights a different drug pair using special tags to mark the two targeted drugs.

- **[e1]** before and **[/e1]** after the first targeted drug.
- **[e2]** before and **[/e2]** after the second targeted drug.

For example, from the original sentence:

> "dose-time effects of competitive displacement of DRUG0 by DRUG1 following oral and intravenous administration, the effect of various doses of DRUG2"

We extract the following three instances by tagging the target drugs.

- **Instance 1 (target drugs are drug0 and drug1)**: "dose-time effects of competitive displacement of [e1] drug0 [/e1] by [e2] drug1 [/e2] following oral and intravenous administration, the effect of various doses of drug2."
- **Instance 2 (target drugs are drug0 and drug2)**: "dose-time effects of competitive displacement of [e1] drug0 [/e1] by drug1 following oral and intravenous administration, the effect of various doses of [e2]drug2 [/e2]."
- **Instance 3 (target drugs are drug1 and drug2)**: "dose-time effects of competitive displacement of drug0 by [e1] drug1 [/e1] following oral and intravenous administration, the effect of various doses of [e2] drug2 [/e2] ."

For each instance, we generate two different representations:

- One version retaining the explicit drug tags `[e1]`, `[e2]`.
- A normalized one where the tags are replaced with abstract placeholders:
  - `[e1]. [/e1]` → `DRUG_E1`
  - `[e2]. [/e2]` → `DRUG_E2`
  - All other drug mentions (e.g., `DRUG+`) → `DRUG_E`

  For example, the sentence:

  ```
  [e1] drug1 [/e1] can be bad if used with [e2] drug2 [/e2] or drug3
  ```

  will be transformed into:

  DRUG_E1 is bad when combined with DRUG_E2 or DRUG_E

The motivation for generating the second representation will be discussed in subsequent sections.

5.1.4 Imbalances Dataset

Imbalanced data is a well-known problem in ML. It is characterized by having significantly more instances of certain classes than others. Since rare instances appear infrequently, classification rules that identify these minority classes tend to be rare, undiscovered, or ignored; as a result, test samples from the minority classes are misclassified more often than those from the majority classes. This issue is particularly prevalent in medical datasets, as noted in [1]. In our case, the class imbalance can be seen in the statistics presented in Table 2.4. It occurs in two ways:

- The number of negative instances in the training set is almost six times greater than the number of positive instances.
- The class `"int"` is severely underrepresented, with only 188 instances.

To solve this problem, we constructed a less imbalanced corpus by removing some negative instances. This was done by using manually-formulated rules proposed by other researchers, defined in [36, 80]. The following rules were applied to remove negative instances:

- If two drugs in a sentence have the same name.
  Example: "Pharmacokinetic properties of abacavir were not altered by the addition of either **lamivudine** or zidovudine or the combination of **lamivudine** and zidovudine."
  In this case, interactions between drugs with the same name (e.g., between the first and second occurrence of lamivudine or between the first and second zidovudine) are removed.
- If one drug is an abbreviation, acronym, or a special case of the other,

  To detect abbreviations, we used regex rules, and synonyms were detected by using the Drugbank vocabulary dataset, which contains more than 17000 drugs with their details, including their synonyms.
- If the two drugs appear in the same coordinate structure
  Example : "To investigate the effects of antimicrobial combinations of GL with four kinds of antibiotics (ampicillin, cefazolin, oxytetracycline, and chloramphenicol), the FICI was determined by assay for each strain."

  Here, the drugs ampicillin, cefazolin, oxytetracycline, and chloramphenicol are in the same coordinate structure. Therefore, any interaction between them is removed.

Table 3.2 summarizes the dataset statistics before and after filtering.

Corpus	**Training set**		**Test set**	
	Before	**After**	**Before**	**After**
Pairs	27,774	18,889	5,716	3,959
Positive DDIs	4,018	3,973	979	968
Negative DDIs	23,756	14,916	4,737	2,991
Mechanism	1,318	1,309	302	296
Effect	1,685	1,659	360	355
Advice	826	818	221	221
Int	189	187	96	96

Table 3.2: Data statistics before and after negative filtering

5.2 Model Architecture

DDI extraction is a challenging issue that cannot be addressed through semantic similarity alone or superficial co-occurrence. In biomedical literature, critical relational signals may be embedded in syntactic constructions or expressed through domain-specific lexical

preferences. For this reason, we believe that DDI extraction depends on the joint modeling of two interrelated views:

- **Semantic** meaning to capture contextual relationships.
- **Syntactic** structure to reveal grammatical dependencies.

The proposed model operationalizes this intuition into practice by combining these two views into a unified hybrid architecture. Instead of focusing solely on pre-trained language models, we emphasize the unique contributions of each component that provides a singular perspective on the data:

- **Semantic view:** We leverage *BioBERT-base (PubMed)* as our sentence and token-level contextualized encoder. To make it more suited to the unique linguistic properties in the DDI2013 dataset, we performed fine-tuning using this dataset, following standard methodologies used in association with biomedical relation extraction [54].
- **Syntactic view:** We parse sentences into dependency trees that express grammatical relations (such as subject, object, and modifier) that can link drugs in not immediately apparent ways from word order alone. We then convert the tree into a graph, and to process this graph, we employ a **GAT network**. The GAT is learned so that it emphasizes the most relevant syntactic pathways, beneficial for discovering long-distance or non-obvious relationships between entities. This syntactic feature supplements BioBERT by making structural dependencies between drugs more overt and learnable.

It has been shown previously that static and contextual embeddings have complementary benefits in specialized domains [6], and some methods use Word2Vec embeddings to initialize representations for drug mentions that are not adequately captured by transformer-based models [40]. To combine these different types of information, we consider two fusion approaches: (1) simple concatenation and (2) cross-attention, which allows for a more refined alignment between the GAT output and the BioBERT [CLS] embedding. The motivation for this attention-based fusion comes from methods used in cross-view learning [78]. The final joint representation is passed through a feed-forward layer and a classifier to predict the DDI type. As shown in Figure 3.8, each component is explained in detail in the following subsections.

5.2.1 Building the Dependency Parsing Graph

We apply dependency parsing on the normalized version of each sentence (e.g., using `DRUG_E1`, `DRUG_E2`, `DRUG_E`) instead of the original drug mentions. This transformation guarantees that special annotation tokens do not influence dependency parsing or the generation of Word2Vec embeddings. As a result, it produces cleaner, more stable dependency structures that are better suited for downstream graph construction.

We use the StanfordNLP dependency parser to generate a syntactic tree, which is converted into a directed graph, where:

- Every node is a token.
- Each edge is a directed pair `(parent , child)`, representing the syntactic head relationship.

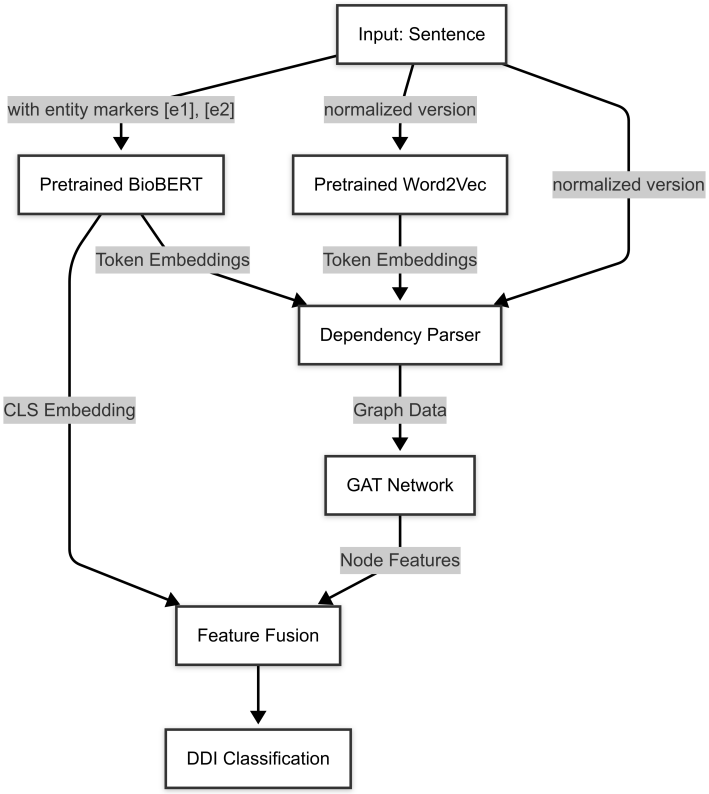


Figure 3.8: Architecture of the proposed Model for DDI relation extraction

The output is an edge list, where each edge is represented as a (parent, child) tuple, which is far more memory-efficient than an adjacency matrix. It enables scalable GAT-based processing without requiring the embedding of the edge types.

For example, consider the following sentence:

"DRUG_E1 inhibits platelet aggregation caused by DRUG_E2."

A dependency parse yields the following child-parent pairs:

- (inhibits, DRUG_E1)
- (inhibits, aggregation)
- (aggregation, platelet)
- (aggregation, caused)
- (caused, by)
- (by, DRUG_E2)

These can be expressed as an edge list:

```
edge_index = [[1, 1, 3, 3, 4, 5],
[0, 3, 2, 4, 5, 6]]
```

Each pair (i, j) in the matrix denotes a directed edge from token i (parent) to token j (child), reflecting the dependency relationship between them. For example:

- Token at index 1 (`inhibits`) → index 0 (`DRUG_E1`)
- Token 1 → 3 (`inhibits` → `aggregation`)
- Token 3 → 2 (`aggregation` → `platelet`)
- Token 3 → 4 (`aggregation` → `caused`)
- Token 4 → 5 (`caused` → `by`)
- Token 5 → 6 (`by` → `DRUG_E2`)

5.2.2 GAT Network

We employ a GAT Network to encode the graph derived from dependency parsing. Unlike standard GNNs, which uniformly aggregate features from neighbor nodes, GAT uses attention coefficients to allow the model to assign varying weights to each neighbor when passing messages. By doing so, the network can better learn which neighboring tokens contribute the most to learning beneficial representations [68].

Each node (token) is represented with a 1068-dim feature vector consisting of:

- 768-dim BioBERT contextual embedding (token-level outputs),
- 300-dimensional Word2Vec embeddings trained on the full

This combined representation enables the graph network to incorporate both global semantic and local lexical information.

The graph edges are constructed based on token-to-token dependencies, but we do not utilize the relation labels (e.g., `nsubj`, `dobj`). Instead, we treat each edge as a structural connection between nodes. GAT propagates information throughout the graph using multiple attention heads, augmenting node representations with context-dependent information. After the message-passing step, global pooling is applied to obtain a fixed-size graph-level embedding. This embedding is concatenated with the BioBERT `[CLS]` representation for final DDI classification.

5.2.3 Feature Fusion

To integrate syntactic (GAT-derived) and semantic (BioBERT-derived) features, we explore two strategies:

- **Concatenation:** GAT-derived features and the BioBERT `[CLS]` embedding are concatenated to form a joint feature vector.
- **Cross-view attention:** we employ a `MultiViewCrossAttention` module (Figure 3.9). In this setting, node features computed from GAT are used as **queries**, and BioBERT-derived `[CLS]` embedding as both **key** and **value**. This attention mechanism enables more advanced alignment of the two representations, so that the model can choose to selectively attend to informative substructures of the graph.

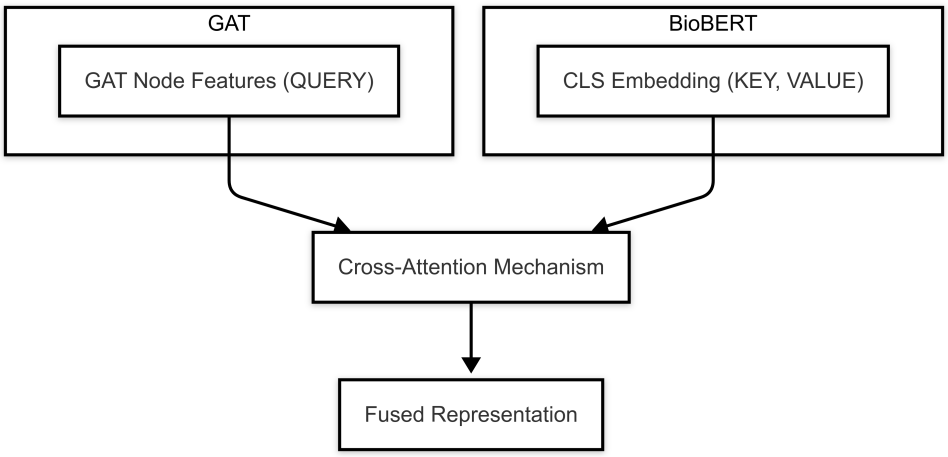


Figure 3.9: Cross-attention mechanism using GAT node features as queries and BioBERT `[CLS]` as keys and values

5.2.4 Classification

The integrated representation is then passed to a fully connected classification layer, which predicts the DDI type. To further address dataset imbalance, we employ focal loss as the loss function, as it has proven effective in handling such issues.
Focal Loss is a loss function specifically designed to address the class imbalance in classification tasks [35]. We chose to use it because, even after applying negative filtering, class imbalance remained a significant issue. It works by down-weighting well-classified examples, which allows the model to focus more on hard, misclassified instances. It is defined as:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t) \tag{3.1}$$

where :

- p_t is the model's estimated probability for the true class.
- α_t is a weighting factor for class t.
- γ is the parameter that adjusts the rate at which easy examples are down-weighted. It is in the range [0, 6]; a value of 5 means the model will focus more on harder instances, while a value of 0 means it will also focus on easy instances.

6 Complete DDI Extraction Workflow

The DDI extraction pipeline analyses an input sentence to identify drug entities and classify potential interactions between them. This pipeline involves a series of sequential steps: text preprocessing, DNER, entity blinding, instance generation, negative instance filtering, and DDI classification. Each step is carefully designed to ensure the precise identification and classification of DDI.
The pipeline operates as follows:

1. **Text Preprocessing**: The input sentence is converted to lowercase for normalisation.

2. **DNER**: The sentence is fed into a fine-tuned BioBERT-large model (BioBERT-large-cased-v1.1) to recognize drug entities. Each token is assigned a BIO tag (`O`, `B-DRUG`, or `I-DRUG`), as explained in Section 4.

3. **Instance Generation**: An instance for every unique pair of drug entities in the sentence is created, as explained before in Section 5.1.3 .

4. **Negative Instance Filtering**: Rule-based filtering is applied to remove negative instances as explained in Section 5.1.4.

5. **DDI Classification**: The filtered drug pairs are fed into our hybrid proposed model, which integrates contextual, semantic, and syntactic information. The model then predicts the interaction type for each drug pair.

Figure 3.10 shows the pipeline for the sample sentence: *"Tetracycline and broad-spectrum antibiotics can reduce the efficacy of contraceptives."*

7 Conclusion

In this chapter, we introduced our proposed solution by describing the chosen dataset and providing a detailed explanation of the architectures for both the DNER model and the DDI relation extraction model.

In the following chapter, we will discuss the tools used for implementation and development, as well as the testing procedures and the results obtained.

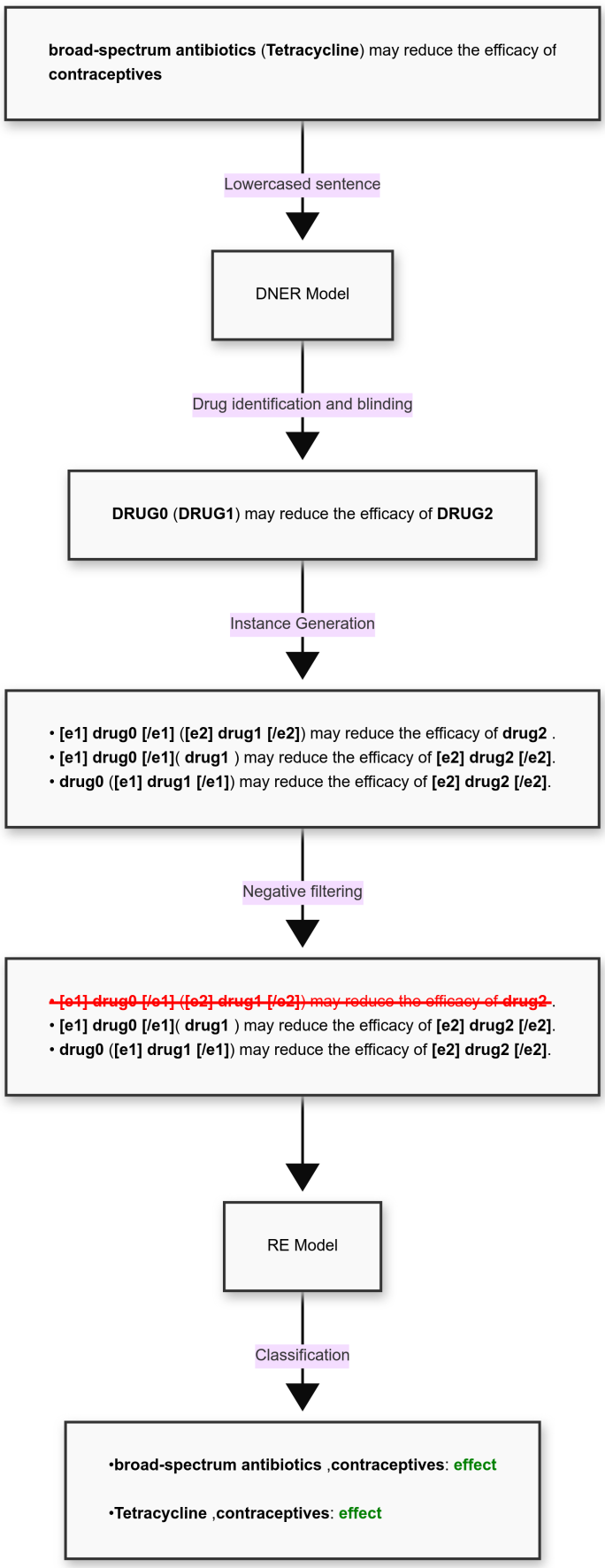


Figure 3.10: Full pipeline for DDI extraction.

CHAPTER 4

EXPERIMENTS AND RESULTS

1 Introduction

This chapter presents the implementation details, conducted experiments, and attained results. It starts with describing the experimental setup and then describes the methods and techniques used for enhancing the performance of both the DNER and RE models. Additionally, it presents the best results for both models and finishes with the demonstration of the application interface developed.

2 Experimental Environment

2.1 Hardware Specification

To evaluate our work, we conducted multiple tests on different architectures, features, and hyperparameters. To properly evaluate these tests, we utilized Google Colab, a platform for running Python code in the cloud, and a local machine with the following specifications:

- **Google Colab**
  - RAM: 12.7 GB
  - GPU: NVIDIA T4 GPU 15 GB
- **Machine 1**
  - CPU: AMD Ryzen 7 3700X (16 cores) @ 3.600 GHz
  - RAM: 16 GB
  - GPU: NVIDIA GeForce RTX 2060 SUPER 8 GB

2.2 Software Specification

Following the hardware details, this subsection covers the key software languages, libraries, and tools used in the project.

- **Python**: a high-level, interpreted, general-purpose programming language known for its clear syntax and code readability. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is widely used in scientific computing, data analysis, artificial intelligence, web development, and automation due to its extensive standard library and active ecosystem [59].

- **Pytorch**: an open-source deep learning framework used to build neural networks, combining the machine learning (ML) library of Torch with a Python-based high-level API. Its flexibility and ease of use, among other benefits, have made it the leading ML framework for academic and research communities [59]

- **Other Tools and Libraries**:

Tool	**Description**	**Use Case**	**Link**
NumPy	Numerical computing with support for arrays, matrices, and linear algebra.	computing, array and matrix manipulation	Link
Optuna	Hyperparameter optimization framework with efficient search algorithms.	Hyperparameter tuning	Link
Torch Geometric	Deep learning library for geometric (graph) data using PyTorch.	GAT	Link
Gensim	Library for unsupervised topic modeling and natural language processing.	Word embeddings (word2vec)	Link
Matplotlib	2D plotting library in Python for high-quality figures.	Visualizations and plots	Link
lxml	High-performance XML and HTML processing library.	Parsing, creating XML files	Link
Pandas	Fast, flexible data analysis and manipulation library.	Data cleaning, CSV handling	Link
spaCy	Industrial-strength NLP library with fast processing	Tokenization , POS tagging , dependency parsing	Link
Scikit-learn	ML library with simple and efficient tools.	Classification, regression, clustering	Link
Transformers	Hugging Face library for pretrained transformer models.	Fine-tuning large language models	Link

Continued on next page

Tool	Description	Use Case	Link
StanfordNLP (stanza)	NLP toolkit from Stanford with neural pipeline for 66+ languages.	dependency parsing	Link
FastAPI	Web framework for building APIs with Python 3.6+ based on type hints.	Building RESTful API	Link
React.js	A JavaScript library for building interactive user interfaces, maintained by Meta.	Frontend web development	Link
NLTK	A Python library for natural language processing and text analysis.	NLP	Link

3 Evaluation and Analysis

3.1 DNER Evaluation

To select the best BERT variant model, we tested many variants, including ClinicalBERT, BlueBERT, SciBERT, and BioBERT. Since reporting a single performance score is insufficient for comparing non-deterministic approaches [49], we tested each model five times and selected the best one based on the average. Table 4.2 shows the average score of each model.

Model	F-score
BioBERT Base	89,98%
SciBERT	89,66%
BioBERT Large	91.06%
ClinicalBERT	89.87%
BlueBERT	90.48%

Table 4.2: Sample comparison of BERT-based models for DNER

As shown in Table 4.2, BioBERT Large outperformed the other variants. Based on these results, we selected BioBERT Large as the baseline model for all remaining experiments. After selecting BioBERT as our baseline model, we performed multiple experiments to optimize the preprocessing pipeline. Table 4.3 illustrates the effect of including versus excluding stopwords during preprocessing.

Method	F-score	Δ
with STOP WORDS	91,4%	-
without STOP WORDS	89%	-2.4%

Table 4.3: Effect of stop word removal on BioBERT performance

As we can see from Table 4.3, removing stopwords caused a notable decrease in performance (-2.4%), suggesting that so-called irrelevant words could have useful contextual

information for biomedical named entity recognition. Therefore, we chose to retain stopwords throughout all the following preprocessing steps.

3.1.1 Hyperparameter Tuning

To further enhance the performance of the DNER model, we conducted hyperparameter optimization using Optuna. Due to the computational cost of fine-tuning BioBERT large, we limited the search to five trials per study. Despite the small number of trials, this still enabled us to discover good hyperparameter configurations.
We defined a search space over several key parameters:

- **Learning Rate** : $[1e{-}6, 1e{-}4]$.
- **Batch Size**: [4, 8, 16].
- **Weight Decay**: $[0.0, 0.1]$

Despite the low number of trials, Optuna managed to identify a well-performing configuration, shown in Table 4.4. Notably, this configuration was selected based on the best mean F1-score across evaluation runs.

Hyperparameter	**Best Value**
Learning rate	5×10^{-5}
Weight decay	0.01
Batch size	16

Table 4.4: Best hyperparameters found for DNER

As shown in Table 4.5 This configuration achieved a macro F1-score of **91.42%**, confirming the effectiveness of even a small hyperparameter search when the space is well-defined.

Configuration	**macro-F1-score**
Baseline	**91.06**%
Optuna (Best)	**91.42**%

Table 4.5: Performance comparison between baseline and Optuna-tuned configuration

3.1.2 Model Results

The Best and final model was built using biobert_large_v1.1, trained without removing the stop words; it achieved a macro F1-score of **91.42%**. The complete set of evaluation metrics is highlighted in Table 4.6.

Metric	**Score**
Accuracy	96.95%
Precision	88.32%
Recall	94.98%
F1 Score	91.42%

Table 4.6: Evaluation metrics for the DNER model

To further assess model performance, Figure 4.1 presents the confusion matrix, while Figure 4.2 illustrates the Area Under the Curve (AUC) for each class.

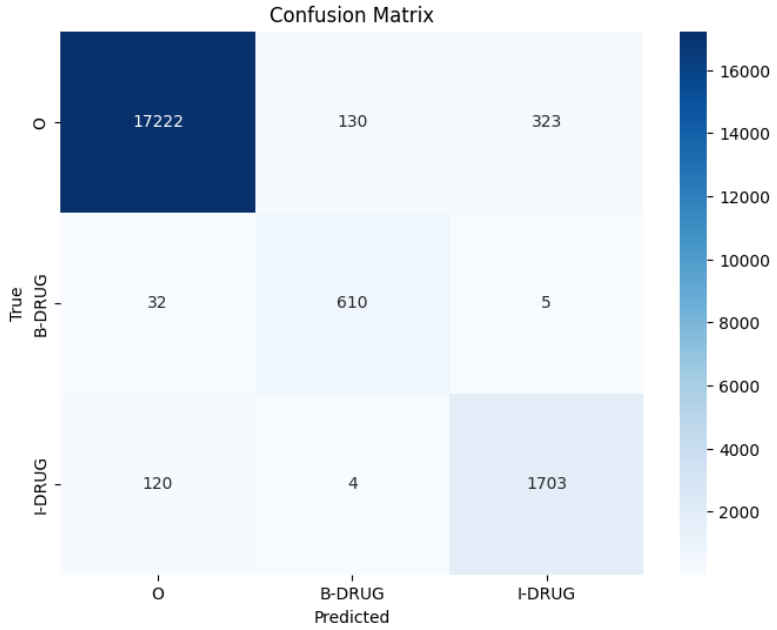


Figure 4.1: Confusion matrix for the DNER model

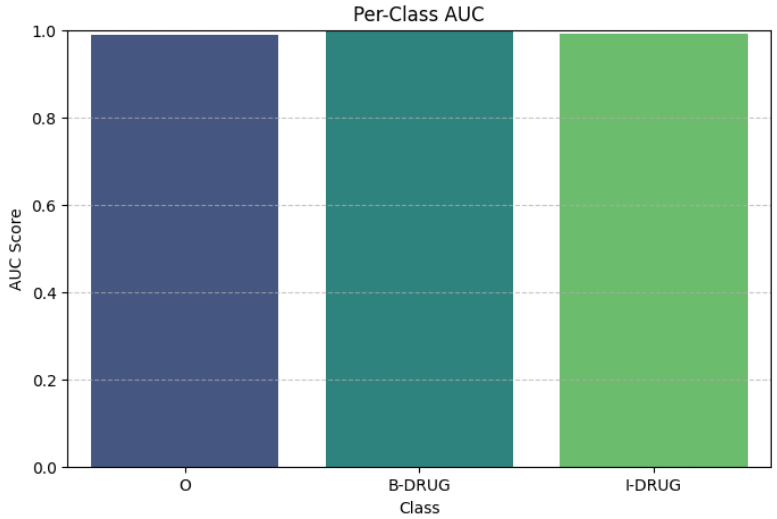


Figure 4.2: Class-wise AUC scores for the DNER model

As shown in Figure 4.1, the model performs very well, especially for the "O" class with 17,222 correct. It also performs well on "I-DRUG" (1,703 correct) and "B-DRUG" (610 correct), with good recall and precision for the most common classes. There is some confusion, e.g., 120 examples of "I-DRUG" are mistakenly classified as "O", likely due to class imbalance. Figure 4.2 supports these findings because all classes have AUC values close to 1. This indicates that the model is generally effective at distinguishing between different entity types, although it still makes some minor errors.

3.2 RE Evaluation

Similar to the DNER task, we compared various BERT variants such as SapBERT, ClinicalBERT, SciBERT, and BioBERT, each trained five times, and selected the best results based on their averages. Table 4.7 shows the average macro F1-scores for the Re task. As shown in the table, BioBERT PubMed outperforms the other variants, which is why we chose to use it for the remaining experiments.

Model	**macro-F1-score**
BioBERT PubMed	77.5%
SciBERT	75,69 %
SapBERT	76,87 %
ClinacalBERT	73.5 %

Table 4.7: Sample comparison of BERT-based models for RE

3.2.1 Handling Class Imbalance

To further address the data imbalance problem, we conducted experiments using several techniques.

Negative Filtering
As explained in the earlier chapter, negative filtering involves selectively removing negative instances to address the data imbalance problem. Table 4.8 highlights the impact of negative filtering on the performance. We note that all experiments were conducted using BioBERT alone.

Metric	**Without Filtering**	**With Filtering**	**ΔF1**
Macro F1	75.12%	79.41%	+4.29%
F1 - None	96.56%	95.48%	-1.8%
F1 - Effect	74.69%	77.01%	+2.32%
F1 - Mechanism	76.44%	82.29%	+5.85%
F1 - Advice	83.63%	83.04%	-0.59%
F1 - Int	44.30%	59.21%	+14.91%

Table 4.8: Impact of Negative Filtering on macro F1-scores for DDI Extraction

As shown in the table, this technique has led to improvements in overall performance across nearly all classes, as well as an increase in the general macro F1-score. This can be attributed to the fact that many of these classes were previously overshadowed by the large number of 'None' instances. It is also worth mentioning that some classes showed a decrease in performance, notably 'None' and 'Advice'. The drop in the 'None' class is expected, as many negative instances were filtered out. As for 'Advice', this can be explained by the fact that, as discussed before, negative filtering also removed some positive instances, potentially including important 'Advice' examples.

Data Augmentation
We applied data augmentation techniques from [70], known as Easy Data Augmentation

(EDA). These techniques randomly perform one or more of the four operations on each sentence:

- **SR (Synonym Replacement):** Randomly choose n non-stop words and replace each with a random synonym.
- **RI (Random Insertion):** Find a synonym of a random non-stop word and insert it at a random position. Repeat this n times.
- **RS (Random Swap):** Randomly choose two words and swap their positions. Repeat n times.
- **RD (Random Deletion):** Remove each word in the sentence with a probability p.

EDA has shown promising improvements when applied to classes with fewer than 500 instances, making it a good fit for the `int` class.
We used the recommended parameters from the original paper, except for the `num_aug` parameter, which controls the number of augmented versions created per sentence. We set it to 9, meaning each instance could have up to nine augmented versions. We also made some modifications in the implementation; for example, in our case, it didn't make sense to delete, swap, or replace drug entities randomly, so those were excluded from augmentation.
After augmentation, the number of Int pairs ranged between 1400 and 1683. This is because the operations were applied randomly, and some sentences were too short to produce all nine versions.
Table 4.9 presents the F1 scores for each class, comparing performance with and without augmentation when using only the BioBERT model.

Metric	**without Augmentation**	**with Augmentation**	**ΔF1**
Macro F1	79.41%	79.40%	-0.01%
F1 - None	95.48%	95.48%	0%
F1 - Effect	77.01%	77.01%	0%
F1 - Mechanism	82.29%	82.26%	-0.04%
F1 - Advise	83.04%	83.04%	0%
F1 - Int	59.21%	59.20%	-0.01%

Table 4.9: Comparison of F1 Scores with and without Augmentation

As shown in the table, there were no improvements in performance. Classes like Mechanism and Int showed a drop in results. This can be attributed to the fact that, as explained in Peculiarities of Medical Language 3.1, Medical texts are unique and require careful handling. These types of augmentations can change important parts of the text or replace key terms, which makes them not useful and even harmful in this context.

Focal Loss
As explained in the earlier chapter, focal loss is a special loss function used to address the data imbalance problem. We compared three strategies for setting the α_t values:

- **Randomized Weighted:** Assigns random weights to each class, biased toward underrepresented ones (e.g., the `int` class).

- **Normalized Inverse Frequency:** Weights are computed using:

$$\alpha_i = \frac{\sum_{j=1}^{N} \text{Count}_j / \text{Count}_i}{\sum_{k=1}^{N} \sum_{j=1}^{N} \text{Count}_j / \text{Count}_k} \tag{4.1}$$

  This gives higher weights to underrepresented classes while ensuring $\sum \alpha_i = 1$. Inspired by [55].

- **macro F1-Based Min-Max Normalization:** Weights are calculated from inverse F1 scores (i.e., $1 - \text{F1}_t$). For each class t, the inverse F1 score is normalized by the minimum and maximum inverse F1 values over all classes:

$$\alpha_t = \alpha_{\min} + \frac{(1 - \text{F1}_t - \min)}{\max - \min} \cdot (\alpha_{\max} - \alpha_{\min}) \tag{4.2}$$

  here, $\min = \min_t(1 - \text{F1}_t)$ and $\max = \max_t(1 - \text{F1}_t)$. Normalization is performed so that classes with bad performance (low F1, high inverse F1) get higher weights. We use the experimental setting $\alpha_{\min} = 0.5$ and $\alpha_{\max} = 1.5$.

Table 4.10 includes the specific α_t values used in each strategy and their performance.

Methods	α_t	**Macro F1-score**	**ΔF1**
Randomized Weighted	0.3,0.7,0.7,0.7,0.9	80.28%	–
Normalized Inverse Frequency	0.008,0.07,0.09,0.15,0.66	79.74%	-0.54
Macro F1-Based Min-Max	0.5,1.04,0.86,0.826,1.5	81.19%	+0.91

Table 4.10: Macro F1 comparison across α_t strategies.

Table 4.10 summarizes the macro F1-scores obtained with different α_t weighting approaches in the focal loss formulation. In addition to performance, the absolute values of α_t employed by every strategy are reported, along with their corresponding relative improvements (deltas) over the baseline.
The baseline approach, Randomized Weighted, achieved an 80.28% macro F1-score. When replacing them with Normalized Inverse Frequency, which assigns more weight to infrequent classes according to frequency-based priors, the macro F1-score decreased to 79.74%, a -0.54% drop in performance. This suggests that class frequencies alone might be insufficient to identify class difficulty or importance in imbalanced biomedical data.
On the contrary, Macro F1-Based Min-Max worked best with a macro F1-score of 81.19%, a +0.91% improvement over the baseline. It computes α_t dynamically from class-wise validation performance, thus giving priority to underrepresented and underperforming classes. Its accommodation for balancing both class frequency and model performance makes it a superior bias-alleviating strategy in this context.
Among all the approaches that were tested, Macro F1-Based Min-Max gave the best balanced and efficient weighting that led to improved generalization across classes.

3.2.2 Graph-Based Representations

To incorporate syntactic structure into our model, we experimented with three types of graph representations that rely on dependency parsing. The graphs were introduced into the model as inputs for a GAT Network.

- **List indexes only:** This encoding only records token positions and their embeddings, but no syntactic edge information. The model relies solely on learned embeddings to calculate structural relations.

- **List indexes + dependency edges (unidirectional):** This more expressive format consists of directed dependency edges between syntactically parsed tokens. The edges are indicated both by their links (as indices) and types (nsubj, obj, amod, etc.) stored in scalar edge attributes. Each edge represents a one-way relation from the parent to its child in the dependency tree. This step attempts to model overt linguistic word dependencies.

- **List indexes + dependency edges (bidirectional):** To enable richer contextual interactions, we also tried a bidirectional graph variant, in which each syntactic edge is duplicated in both directions (child → parent and parent → child). This supports message passing between heads and dependents in both directions.

The dependency graphs were generated through the Stanza dependency parser. Each sentence was translated into a directed graph in which nodes are tokens and edges denote grammatical relations such as "subject", "object", or "modifier". In addition to linking the tokens, each edge also has a scalar value denoting the type of syntactic relation. These scalar values were obtained by assigning to each dependency relation label a particular index and then linearly normalizing these indices to the range $(-1, 1)$. A value of zero is assigned as a default when an unseen or unknown dependency label is encountered.

The performance comparison is summarized in Table 4.11.

Graph Representation	**macro-F1-score**	**Δ**
List Indexes Only	81.14%	-
List Indexes + Dependency Edges	81.19%	+0.05%
List Indexes + Dependency Edges (bidirectional)	80.75%	-0.42%

Table 4.11: Comparison of Graph Representations

As shown in Table 4.11, incorporating dependency edge information provided a slight improvement relative to the baseline (List Indexes Only), increasing macro-F1 from 81.14% to 81.15%. This suggests that syntactic dependency edges can provide slightly useful structural information for relation extraction tasks.
However, introducing bidirectional edges decreased the performance to 80.75%, down by 0.42%. Even though bidirectional message passing was intended to enhance contextual flow between heads and dependents, it may have introduced redundant or noisy signals, triggering over-smoothing or vagueness in graph propagation. These results indicate that adding syntactic structure is potentially beneficial, but it is sensitive to graph construction and expressiveness vs. noise trade-off.

3.2.3 Feature Ablation and Analysis

As described in the earlier chapter, we experimented with two ways of feature fusion: concatenation and cross-view attention. Table 4.12 compares the performance of the two fusion methods using the same representative feature combination (Word2Vec + BioBERT Token + BioBERT [CLS]).

Fusion Method	Macro F1-score
Concatenation	78,24 %
Cross-Attention	81.19%

Table 4.12: Comparison between fusion strategies

We have experimented with a number of combinations and observed that cross-attention tended to work better. The results in Table 4.12 reflect the best observed performance for each fusion method. In the cross-attention setup, View 1 is utilized as the query and View 2 as the key and value so that the model can selectively pay attention to the most significant semantic features during fusion. Each feature view is constructed from different embedding sources:

- **Word2Vec**: Pretrained static word embeddings (300 dimensions).
- **BioBERT Token Embeddings**: Contextual embeddings from the last four layers of BioBERT (768 dimensions), extracted for each token.
- **BioBERT [CLS] Token**: The sentence-level embedding from BioBERT's [CLS] token (768 dimensions) from the last four layers.
- **POS Scalar Feature**: A scalar encoding of each token's POS tag. Each tag is first assigned a unique index equal to its position in a predefined list of POS categories (e.g., "ADJ", "ADP",..., "SPACE"). These indices are then linearly normalized to fall between the values$[-1, 1]$. This provides a single scalar value per token that represents syntactic information in a compact way. For instance, when the POS tag is "AUX", the corresponding normalized scalar can be 0.5.

Table 4.13 presents the combinations of these features used as **View 1** (query) and **View 2** (key/value) in the fusion mechanism, along with the corresponding macro F1-score achieved by each configuration.

View 1	View 2	Macro F1-score
Word2Vec	BioBERT [CLS]	80.70%
Word2Vec + POS	BioBERT [CLS]	80.78%
Word2Vec + BioBERT	BioBERT [CLS]	81.19%
Word2Vec + POS + BioBERT	BioBERT [CLS]	81.03%
BioBERT	BioBERT [CLS]	81.09%

Table 4.13: Performance comparison for different feature fusion combinations.

Results in Table 4.13 indicate that adding more complementary features improves model performance. The best performance (81.19%) was achieved by pairing **Word2Vec + BioBERT token embeddings** as View1 (graph input) with **BioBERT [CLS]** as View2. This suggests that the combination of local contextual embeddings (from BioBERT tokens) and pre-trained semantic features (from Word2Vec) provides a richer representation when paired with sentence-level context from the [CLS] token.
Surprisingly, adding the POS scalar feature did not lead to additional improvement and even caused slight damage in some cases. This may indicate that syntactic information encoded as one scalar per token may not be informative enough or may introduce noise when combined with more expressive semantic embeddings.

3.2.4 Hyperparameter Tuning

To further enhance the performance, we used Optuna for hyperparameter tuning. We defined a search space for different basic hyperparameters and experimented with every setting using 10-fold cross-validation, where the macro F1-score was used as the target for optimization. The most important hyperparameters we attempted were initially selected from broad, random ranges. However, by analyzing optimization results, particularly with Optuna's visualization capabilities and trial history, we could identify better subranges for significant parameters (as shown in Figure 4.3).

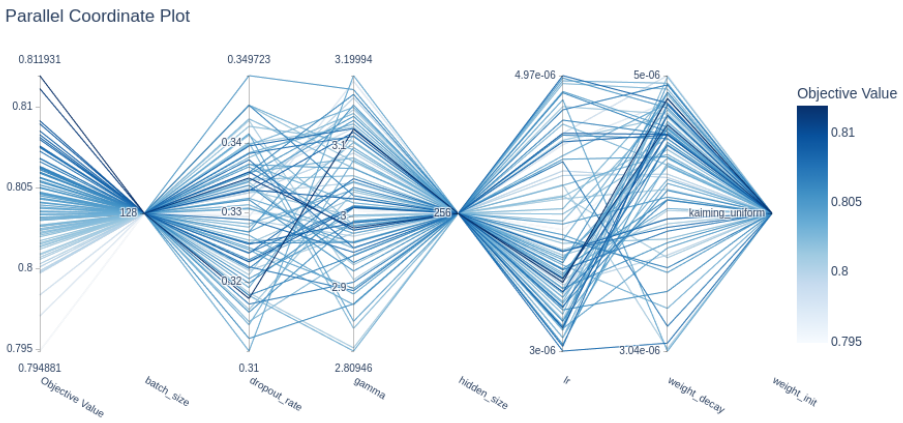


Figure 4.3: Parallel coordinate plot from Optuna showing the influence of key hyperparameters

Here are the parameters that we experimented with:

- **Learning Rate (`lr`)**: Ranged from 1e−5 to 5e−4.
- **Gamma (γ)**:γ is a focusing parameter of the Focal Loss function; it controls the loss contribution of easy examples so that the model concentrates more on hard, misclassified examples.

  During our experiments, we began with a wide range of values γ from 1 to 6. We observed that:

  - $\gamma > 4$: the model began to over-emphasize the high α classes, over-weighting their impact and ignoring lower-α classes, despite not being completely excluded.
  - $\gamma < 2.5$: the model was not placing sufficient emphasis on the important, under-represented classes; it was behaving closer to a normal cross-entropy loss, reducing the intended effect of α in class weighting.

  From these results, we determined that having γ in the range of 3 to 4 was an optimal balance between emphasizing complex examples and respecting the class importance assigned by α.
- **Dropout Rate**: Between 0.1 and 0.6.
- **Weight Decay**: Attempted between [3e-6, 5e-6]

- **Weight Initialization:** Weight initialization is a technique used to initialize the weights of a neural network before training. The goal is to prevent the exploding or vanishing of activation outputs during forward and backward propagation, which can slow down learning and convergence [2]. In our experiments, we attempted multiple initialization strategies, including:

  `kaiming_uniform`, `xavier_normal`, `kaiming_normal`, and `orthogonal`.

- **Hidden Size** and **Batch Size**: Although we originally considered tuning these values, experiments showed that using a `hidden_size` of 256 and a `batch_size` of 128 had the most performant results. As a result, these values were fixed during the final Optuna trials to focus the search on more impactful hyperparameters.

Furthermore, as shown in Figure 4.4, the evolution of objective values over trials highlights the improvement trend across successive Optuna trials, confirming the convergence behavior and effectiveness of the hyperparameter tuning process.

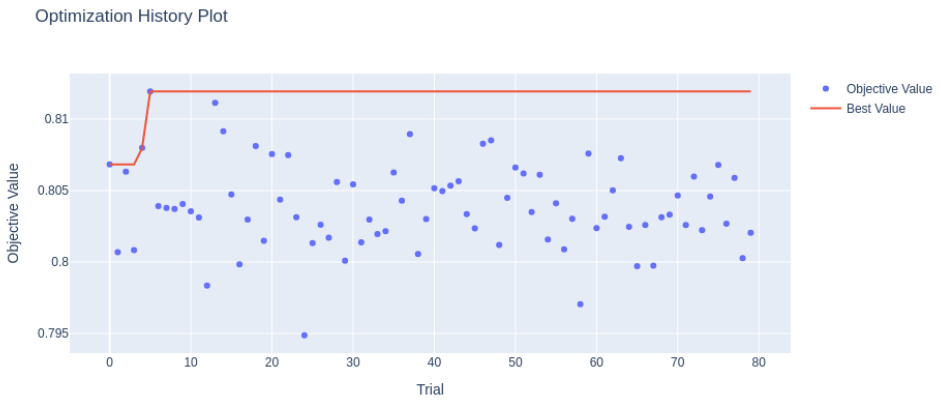


Figure 4.4: Objective values across Optuna trial

Optuna was executed multiple times, each with a maximum of 100 trials. Whereas not included in the Optuna search, we also manually tested some training strategies as an addition to automated optimization:

- **Optimizers:**

  Choosing a suitable optimizer is important for stabilizing training and achieving good generalization. We have experimented with the following optimizers and optimizer combinations:

  - `AdamW`: A variant of Adam with decoupled weight decay, which decouples L2 regularization from gradient-based updates for better generalization.
  - `Lookahead + AdamW`: Interpolates between quick and slow weights, pairing AdamW's fast updates with Lookahead's stabilization technique. This helps to improve the robustness of tasks.
  - `RAdam (Rectified Adam)`: Rectifies Adam's variance in early stages through the use of variance estimates to adjust the learning rate.

- `Lookahead + RAdam`: Hybrid optimizer unifying rectification and update stabilization.
- `Lion`: A memory-efficient optimizer with momentum and sign-based updates, which is designed to be more memory-efficient while being competitive with Adam-like optimizers.

After extensive testing, the overall best performance was achieved with the default `Adam` optimizer as implemented in `torch.optim.Adam`. While older than some of its recent variants like RAdam or Lion, Adam showed the most stable convergence and overall best final performance on our task.

- **Learning Rate Schedulers**: Learning rate schedulers adjust the learning rate throughout training to optimize convergence and generalization. An excessively large constant learning rate would lead to divergence, and a small learning rate would lead to slow learning or convergence into poor minima [53].

  We tried some of the most popular scheduling strategies in our experiments, which are:

  - `ReduceLROnPlateau`, which decreases the learning rate once a monitored metric (e.g., validation loss) plateaus.
  - `Linear Warmup with Decay`, which is the typical method used in transformer models, in which the learning rate linearly increases in initial training steps (warmup) to stabilize initial updates and then linearly decays.
  - Fixed step schedules, in which the learning rate is manually reduced at certain epochs.
  - `Custom Learning Rate Decay (LambdaLR):`

    The best performance was achieved among all learning rate scheduling strategies tried with PyTorch's `LambdaLR` scheduler. The application of this technique gives complete control to the learning rate by providing a custom decay rule through a lambda function.

    Unlike predefined strategies, `LambdaLR` enables specifying a custom learning rate scaling for each epoch, allowing us to fine-tune training dynamics precisely. This flexibility proved beneficial in our case, where standard schedulers did not align well with the model's convergence behavior.

After extensive automated and manual hyperparameter searches, we chose the best-functioning configuration that yielded the best performance on validation folds.
Table 4.14 presents the optimal hyperparameters identified.

Hyperparameter	**Best Value**
Learning rate	$3.404164584946852e \times 10^{-6}$
Gamma	3.1240248614193034
Dropout	0.31757180354538006
Weight decay	$4.792164812040827 \times 10^{-6}$
Batch size	128
Hidden size	256

Table 4.14: Best Hyperparameters Hyperparameter Settings

The model achieved a macro-F1-score of 81.19% with these fine-tuned values, as presented in Table 4.15. This again points towards the significant role played by properly tuned hyperparameters in achieving state-of-the-art results.

Configuration	macro-F1-score
Baseline	**80.60**%
Optuna (Best)	**81.19**%

Table 4.15: Performance comparison between baseline and Optuna-tuned configuration

3.2.5 Ensemble Learning

Ensemble learning is a machine learning technique that combines two or more learners to produce more accurate predictions. In other words, an ensemble model combines several models to produce more accurate predictions than a single model alone [59].
There are different strategies for ensemble learning, and the two common ones are hard voting and soft voting. In hard voting, each model makes a class prediction, and the final output is the majority vote among them. In contrast, soft voting averages the predicted class probabilities from each model and selects the class with the highest average probability.
In our case, we applied ensemble learning at the end of the pipeline using the same model architecture with different hyperparameters. We stored our top-performing 11 models and tested all combinations possible of three, four, five, up to nine models using hard voting to enhance overall performance. We chose the combination with the highest macro-F1 score. Table 4.16 shows the performance comparison between our single best model and ensemble learning (hard voting).

Metric	Single Best Model	Ensemble (Hard Voting)	ΔF1
Macro F1	81,19 %	81.44%	+0.25%
F1 - None	95,77 %	95,89%	+0.12%
F1 - Effect	78,93%	78.65%	-0.28 %
F1 - Mechanism	83,54 %	84,59%	+1.05%
F1 - Advice	85,07%	85.39%	+0.32%
F1 - Int	62,67 %	62.67%	0%

Table 4.16: Comparison between the Single Best Model and Ensemble Learning (Hard Voting)

As shown in the table, the ensemble approach led to an increase in the overall macro-F1 score, with minor improvements across most individual classes. It is also worth noting that the 'Int' class did not show any improvement, while the 'Effect' class saw a decrease.

3.2.6 Overfitting Analysis

BERT-based models are known to be prone to overfitting, particularly if trained on small or unbalanced datasets. In our experiments, we noticed that the model is highly overfitting: the training loss kept on decreasing, while the validation loss initially decreased before it began to increase after some number of epochs.

Figure 4.5 shows the training and validation loss evolution during training, highlighting this pattern of overfitting. Similarly, Figure 4.6 presents macro F1-score vs. epoch.

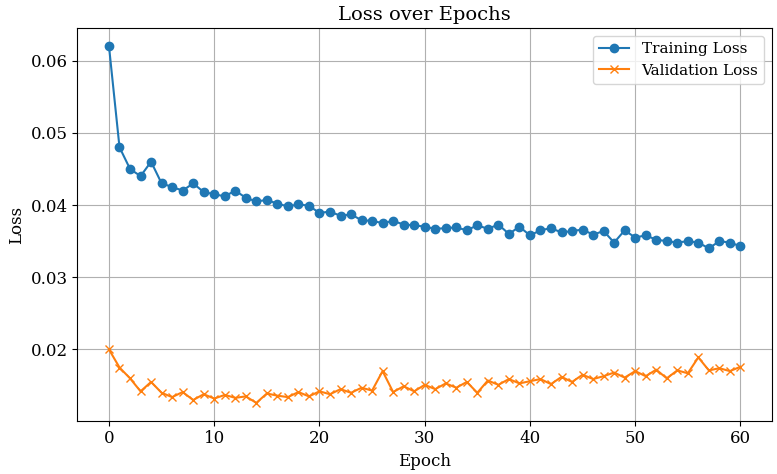


Figure 4.5: Training and Validation loss before regularization

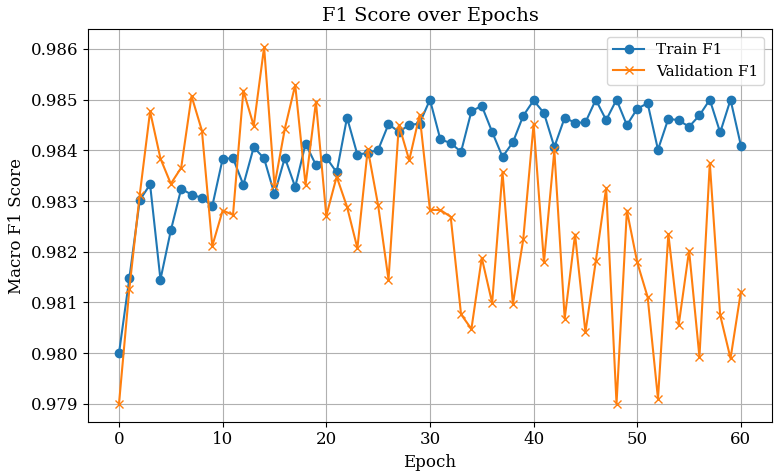


Figure 4.6: Training and validation macro F1-score before regularization

To strengthen the model's robustness against overfitting, we employed the following regularization techniques :

- **Dropout**: we experiment with dropout rate values between $p = 0.1$ and $p = 0.6$ to randomly drop out neurones while training.
- **Layer Normalization**: which is applied for the normalization of activations and promotion of generalization.
- **Early Stopping**: the model stopped when there was no improvement for 30 consecutive epochs.

These strategies helped to reduce the generalization gap between training and validation loss, as illustrated in Figure 4.7; both curves now exhibit similar trends without the previous divergences. The improvement is also reflected in the macro F1-score (Figure 4.8).

These results indicate that the regularization methods improved generalization and reduced overfitting, as evidenced by both the loss curves and the final performance metrics.

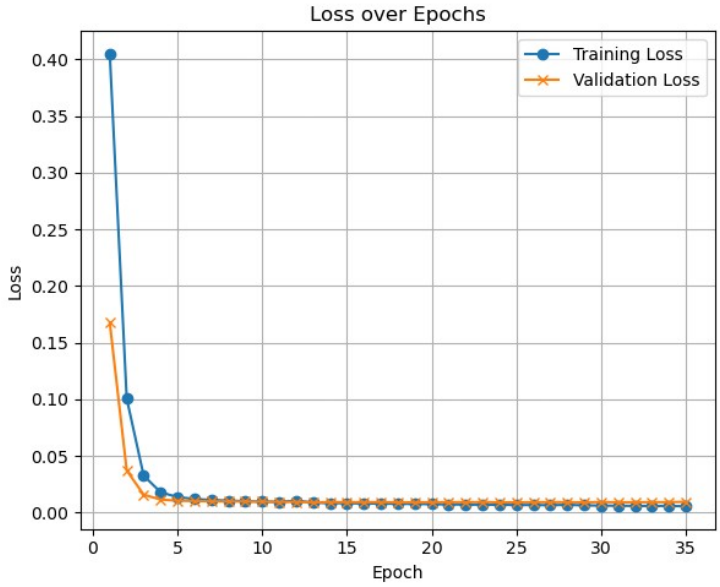


Figure 4.7: Training and validation loss for our proposed model

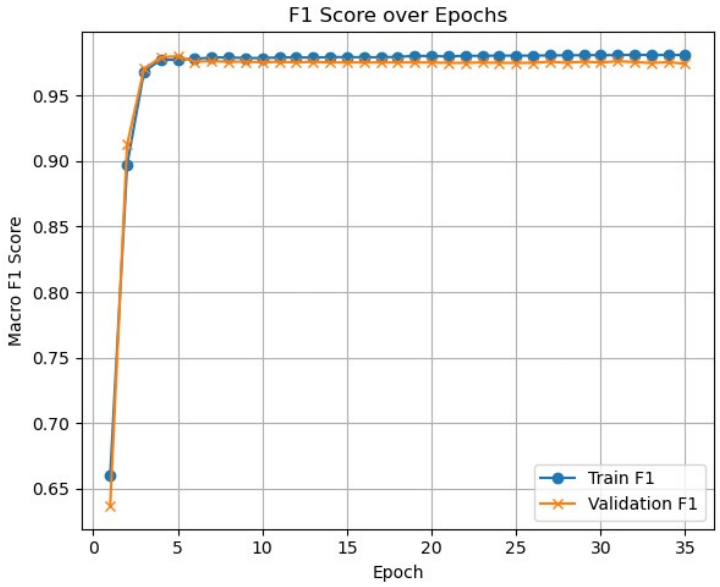


Figure 4.8: Macro F1-score for our proposed model

3.2.7 Model Results

The final model was built using *biobert_v1.1_pubmed*, trained with the focal loss function min-max strategy, and graph data with edge information. For the cross-view features, we combined Word2Vec and BioBERT embeddings as the first view and the BioBERT [CLS] token as the second view. As shown in Section 3.2.5, we applied ensemble learning, which resulted in a slight improvement in the macro F1-score. Table 4.17 compares performance across the different configurations.

- NF: Negative Filter
- Simple Concat : Proposed solution with Simple Concat
- Cross Attn : Proposed solution with Cross Attention
- Ens : Ensemble Learning

Metric	**BioBERT + NF**	**Simple Concat**	**Cross Attn**	**Cross Attn+ Ens**
None	95.48%	94.30%	95.77%	**95.89%**
Effect	77.01%	75.80%	**78.93%**	78.65%
Mechanism	82.29%	80.91%	83.54%	**84.59%**
Advise	83.04%	84.43%	85.07%	**85.39%**
Int	59.21%	55.76%	**62.67%**	**62.67%**
$Precision_{macro}$	83.81%	81.37%	85.73%	**86.79%**
$Recall_{macro}$	76.92%	76.91 %	**78.88%**	78,39%
Macro F1	79.41%	78.24%	81.19%	**81.44%**
Micro F1	91.44%	79.08%	92.07%	**92,27%**

Table 4.17: Performance comparison of BioBERT and proposed methods.

As shown in Table 4.17, all the suggested improvements achieved measurable improvements in performance. Starting from the baseline configuration (BioBERT + negative filtering).
The simple concatenation method was not an improvement upon the baseline when measured in macro F1-score, indicating that naive concatenation of embeddings cannot take advantage of complementary information between views.
In contrast, the cross-attention strategy realized a significant macro F1-score improvement from 79.41% to 81.19%. The majority of classes saw an improvement, especially effect and Int, with better representation of learning for minority and context-dependent classes. It also achieved a higher F1 score for the effect class (78.93%) and a higher overall recall (78.88%).
Finally, ensemble learning usage provided a slight additional lift, and the best macro F1-score was 81.44%. While the improvement over the cross-attention model is slight, the ensemble achieved higher Precision (86.79%) and improved F1 score performance on the mechanism, negative, and advice classes, which means better generalization.The overall recall and the F1 score for the effect class slightly decreased compared to the Cross-Attention strategy
The Int class continues to be the most troublesome, with marginal improvement across all environments, indicating the persistent impact of class imbalance.
In summary, these results substantiate that the utilization of graphs, multi-view embeddings, and ensemble learning produces a more generalizable and precise DDI relation extraction model.
Table 4.18 provides detailed evaluation metrics for each class when using ensemble learning.

Class	Precision	Recall	F1-score
None	95,10 %	96,69%	95,89%
Effect	75,58%	81,97%	78,65%
Mechanism	90,08 %	79,73%	84,59%
Advise	86,18 %	84,62%	85,39%
Int	87,04%	48,96 %	62,67%
Global Macro	86,79%	78,39%	81,44%
Global Micro	92,27%	92,27%	92,27%

Table 4.18: Evaluation metrics per class and global macro average

To further assess model performance, Figure 4.9 presents the confusion matrix.

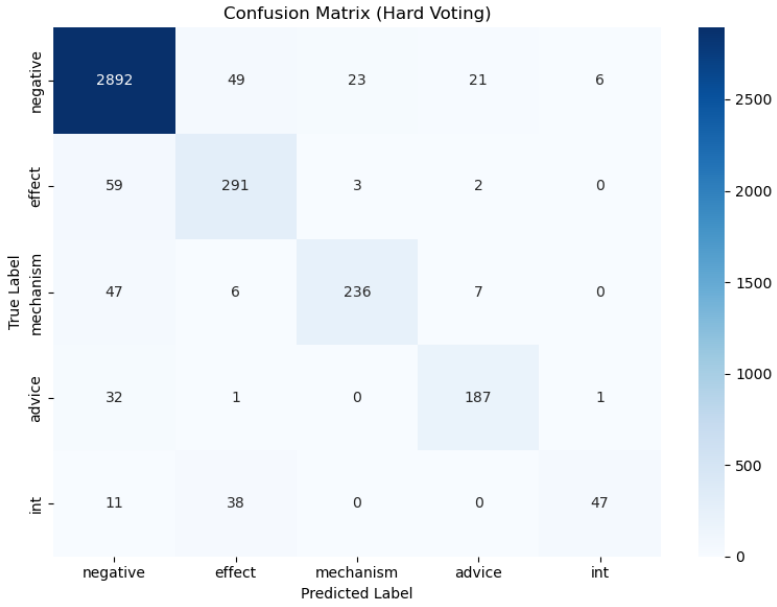


Figure 4.9: RE Confusion Matrix

As shown in the figure, the "int" class had the poorest performance, with approximately 40% of its predictions (38 out of 96 instances) misclassified as "effect". Notably, the "effect" class shows no misclassifications with "int", suggesting distinct class characteristics. This inconsistency may stem from the severe underrepresentation of "int", which has only 187 instances in the training data. Additionally, we observe that "mechanism" was never misclassified as "int", and vice versa, which indicates that the model could clearly distinguish between the two. The strong diagonal performance (2,892 for "negative", 291 for "effect", 236 for "mechanism", 187 for "advice", and 47 for "int") also shows that the model performs reliably for well-represented classes. The class imbalance, especially for "int", remains a persistent problem despite applying several techniques to address it, suggesting that the issue still needs to be addressed.

3.3 End-to-End Architecture Performance

While both DNER and RE models individually achieved strong performance, it is essential to test them together within the full system. This provides a more realistic and fair

assessment of how the models will behave in practice, where they operate jointly. To evaluate this setup, we conducted two main evaluations.

3.3.1 Instance-Level Evaluation

In this evaluation, we treated the task as a binary classification problem at the instance (i.e., drug-pair) level. The goal is to measure how well the pipeline performs at extracting correct drug-drug interaction instances. The evaluation relies on the following definitions:

- **TP**: An extracted instance that also exists in the test set.
- **FP**: An extracted instance that does not exist in the test set.
- **FN**: An instance that exists in the test set but was not extracted by our system.

Table 4.19 provides a detailed breakdown of the instance-level evaluation results.

Metric	**Value**
Precision	86.42%
Recall	93.08%
F1-score	89.63%

Table 4.19: Instance-level evaluation metrics for the full pipeline

Table 4.19 shows instance-level evaluation metrics with all drug pairs treated as individual binary classification instances. The model recorded 3685 TP, 579 FP, and 274 FN. Its high 86.42% precision indicates that the vast majority of predicted interactions were correct, and its 93.08% recall indicates that the majority of true instances of interaction were being detected correctly. The resulting F1-score of 89.63% confirms a high precision-recall balance. Most of the false negatives are likely to be the result of DNER errors, where either or both mentions of the drugs failed to be captured, preventing proper pair construction. False positives are also produced when the model incorrectly identifies an interaction between unrelated entities or when non-drug tokens are tagged as drugs.

3.3.2 Relation Classification on Extracted Instances

After evaluating the pipeline's ability to correctly extract drug interaction instances, we next evaluate its ability to classify the type of interaction of those instances. To do this, we simply evaluate the RE model on the subset of instances that were correctly identified during the extraction phase. This means removing any test instances that were not extracted by the pipeline and computing classification metrics only on the remaining instances. Table 4.20 provides detailed evaluation metrics for each class.

Class	Precision	Recall	F1-score
None	95,25 %	96,65%	95,95%
Effect	75,34%	83,73%	79,32%
Mechanism	90,51 %	80,63%	85,29%
Advise	85,64 %	83,50%	84,56%
Int	85,11%	44,94 %	58,82%
Global Macro	86,37%	77,89%	80,79%
Global Micro	92,29%	92,29%	92,29%

Table 4.20: Evaluation metrics per class and global macro average

To further assess model performance, Figure 4.10 presents the confusion matrix.

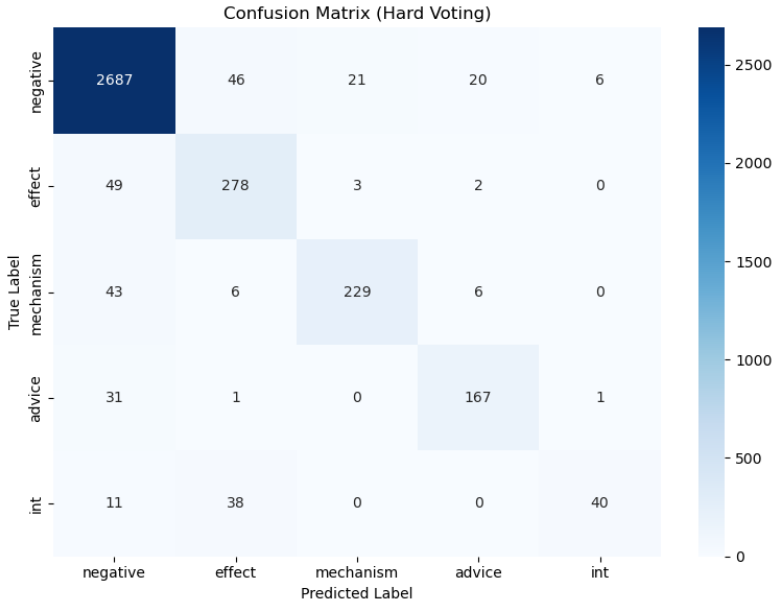


Figure 4.10: Full Pipeline Confusion Matrix

As shown in the figure, the performance of the RE model within the full pipeline is comparatively quite close to its standalone evaluation. The "int" class still performs worst, with approximately 43% of its predictions (38 out of 89 instances) being inaccurately tagged as "effect". And once again, we observe no confusion in the reverse direction: the "effect" class is never misclassified as "int", reinforcing the idea that the model is more confident about recognizing "effect" than "int". Additionally, just like the standalone RE evaluation, we observe no confusion between "int" and "mechanism", suggesting the model successfully separates these two classes.
The overall class-wise performance, as seen by the strong diagonal (e.g., 2,687 for "negative", 278 for "effect", 229 for "mechanism", 167 for "advice", and 40 for "int"), indicates that the integration of DNER does not introduce significant degradation in RE accuracy for well-represented classes. This confirms that the relation extraction module remains stable and reliable in the full pipeline setting. However, the consistently poor performance of "int" underscores the need for additional methods to address class imbalance and improve the detection of rare relations.

4 Application Features and User Interface

To facilitate the use of our two models, it was important to create a graphical interface. In the following, we will present the various functionalities and interfaces of our application.

Welcome Page This page (Figure 4.11) briefly explains the motivation behind the project and showcases the performance of the two models.

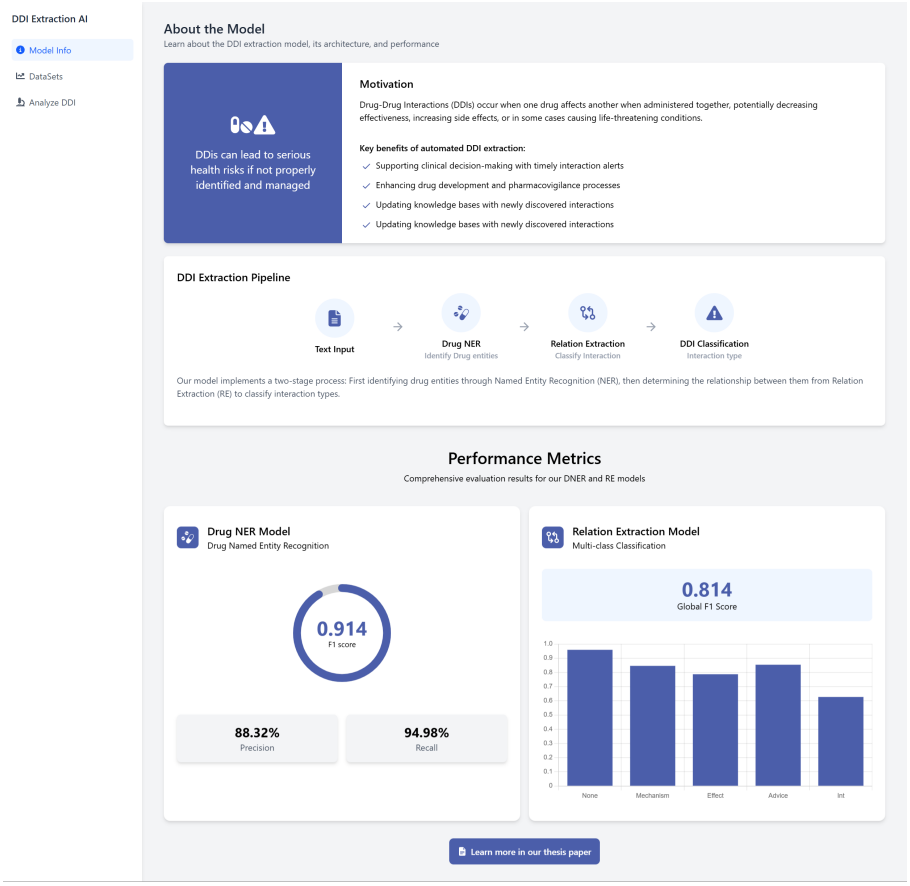


Figure 4.11: welcome Page

Statistics and dashboard This page (Figure 4.12) presents an overview of the DDI2013 Extraction dataset used in our model. It explains the label types, outlines the dataset structure, and showcases examples for each interaction category.

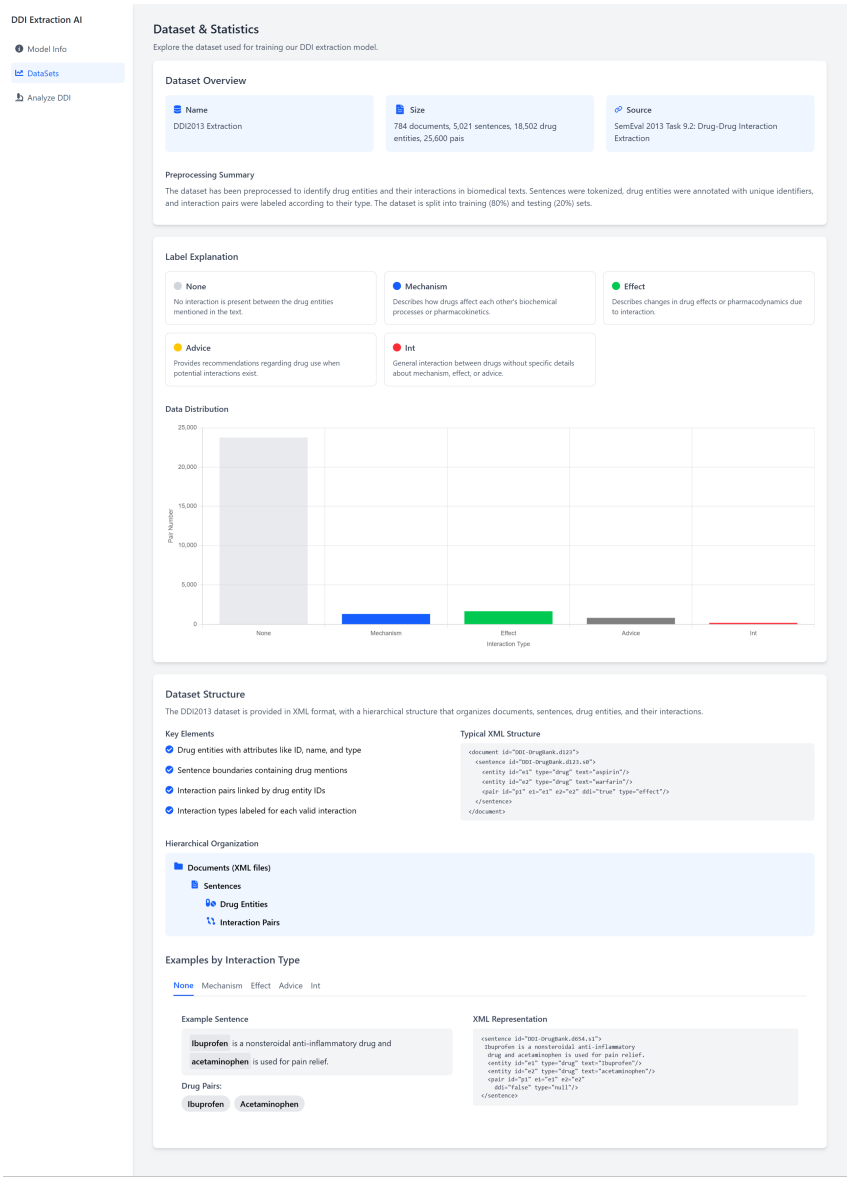


Figure 4.12: Statistics and dashboard

DDI Analyser This page (Figure 4.13) allows users to input sentence DDI pairs along with their corresponding interaction types.

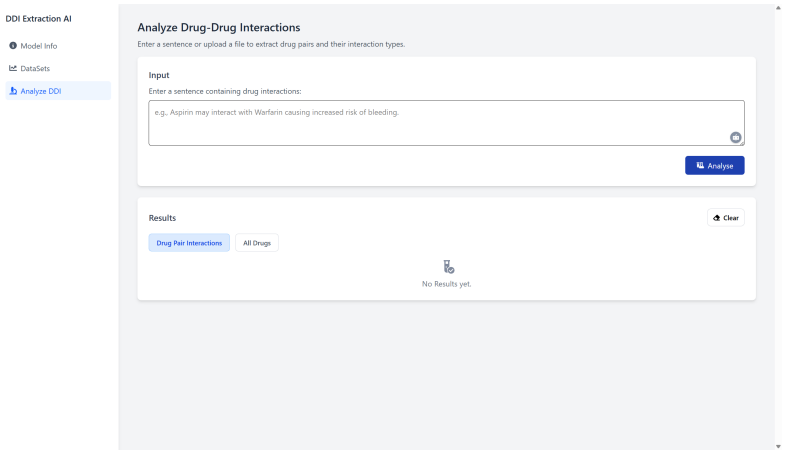


Figure 4.13: DDI Analyser Page

After the sentence is entered, the system extracts all drug entities and drug pair interactions, displaying the results in the tables as shown in Figure 4.14 and Figure 4.15.

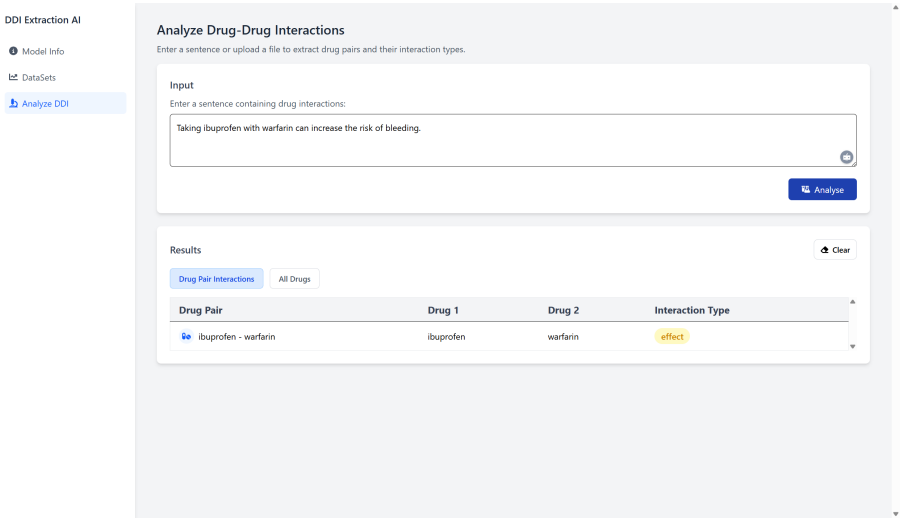


Figure 4.14: DDI Analyser Page (DDI Extracted)

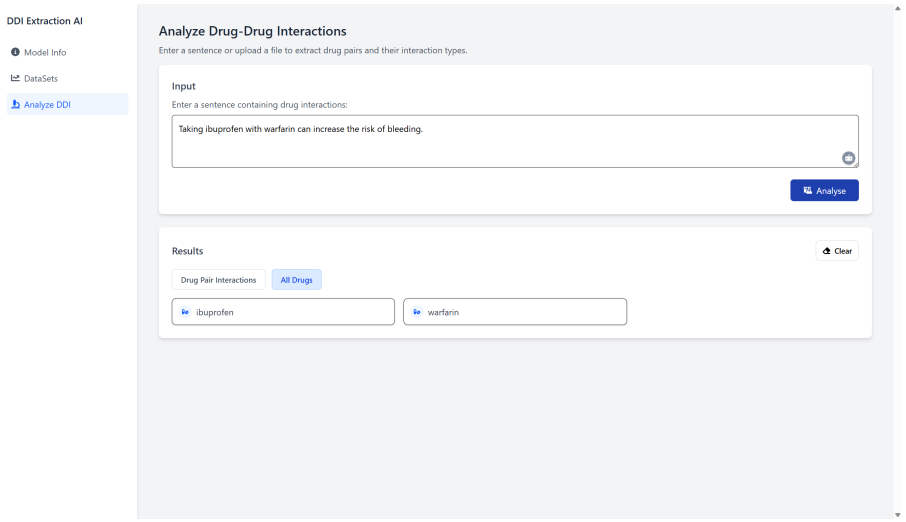


Figure 4.15: DDI Analyser Page (Drug Entities Extracted)

5 Conclusion

In this chapter, we presented the experimental environments, along with the methods and experiments conducted to improve performance. We also described the evaluation metrics and analyzed the results and overall performance. Finally, we presented the application interface (UI) developed as part of this work. Our proposed architecture and overall solution showed a good improvement compared to using BioBERT alone, both in overall performance and in individual classes.

GENERAL CONCLUSION

Throughout this work, we focused on the problem of pairwise DDI extraction. The main objective of this project was to create a self-reliant pairwise DDI extraction system, one that can successfully extract drugs and the relations between drug pairs.
To achieve this, we adopted a modular approach consisting of two separate models: one for DNER and another for RE. This design choice was motivated by an analysis of the current state of the art, which revealed the strengths and limitations of existing systems. We aimed to build upon these insights to develop a more effective and maintainable system.

The proper collection and selection of the dataset was an important step, during which we conducted in-depth research on all available datasets and identified what is considered high-quality and accepted by experts and researchers in the field. The collected dataset was then preprocessed to suit the specific requirements of each subtask. After that, we had to choose the architecture, during which we explored a variety of model architectures and strategies, including fine-tuning transformer-based models and experimenting with graph neural networks. For DNER, we employ BioBERT and fine-tune it to locate drug entities in text accurately. For RE, we integrate BioBERT contextual semantic embeddings with syntactic structures obtained through dependency parsing, which generates graph representations of grammatical relations between tokens. We feed these graphs into a GAT whose nodes are initialized by a mixture of BioBERT token embeddings and Word2Vec embeddings.

The final step consisted of evaluating and validating the proposed solution. To evaluate our models, we mainly focused on the F1 score for DNER and the macro F1 score for the RE model. We obtained an F1 score of 91.4% for DNER and a macro F1 score of 81.44% for RE. These results demonstrate that our approach is capable of effectively extracting both drug mentions and drug pair interactions.

We have shown that our system can extract drug entities and the relationships between drug pairs without human intervention. Through the application we developed, we applied all the knowledge acquired during our training at the university. This project also allowed us to acquire and deepen new concepts in the field of bioinformatics and, more precisely, the concepts of NLP and Deep learning (like LLMs and GNNs).

Future Work: For future perspectives, we plan to introduce the possibility of entering and handling entire articles and files into our application. We also recognize the need to further address the class imbalance problem, particularly for the underrepresented

classes. We can try data augmentation techniques that adapt to the characteristics of the medical texts. It would also be interesting to try the "multitask learning" method, as it seems promising, and also to test the proposed model on different datasets like TAC 2018 and 2019 and perhaps work on multiple DDI extractions.

APPENDIX A

THEORETICAL BACKGROUND

1 POS Tagging Categories

POS tags can be broadly categorised into two types:

- **Universal POS Tags**: These tags are used in the Universal Dependencies (UD) project, which is developing cross-linguistically consistent treebank annotation for many languages. These tags are based on the type of words. E.g., NOUN(Common Noun), ADJ(Adjective), ADV(Adverb) [66]. Table A.1 presents the most commonly used Universal POS tags and their descriptions.

Tag (Abbreviation)	Description
ADJ	Adjective
ADP	Adposition
ADV	Adverb
AUX	Auxiliary
CCONJ	Coordinating conjunction
DET	Determiner
INTJ	Interjection
NOUN	Noun
NUM	Numeral
PART	Particle
PRON	Pronoun
PROPN	Proper noun
PUNCT	Punctuation
SCONJ	Subordinating conjunction
SYM	Symbol
VERB	Verb
X	Other

Table A.1: List of Universal POS Tags

- **Language-Specific POS Tags** Some languages require additional POS tags to capture unique grammatical features beyond the universal set. For example:

- **PRT** (Particle) in the Penn Treebank Tagset for English.
- **NSUFF** (Nominal Suffix) in Arabic, used for possessive pronouns attached to nouns.

2 TF-IDF and Bag-of-Words

- **Bag of words**: A bag-of-words (BoW) model is a simple document embedding technique based on word frequency. It treats text as a "bag" of words, tracking the frequency of each word and converting these counts into numerical values. Instead of considering word order, the model represents a document as a vector where each entry corresponds to the frequency of a specific word from a predefined vocabulary [76].

  To illustrate this concept, consider a short passage from A Tale of Two Cities by Charles Dickens:

  "It was the best of times, it was the worst of times; it was the age of wisdom, it was the age of foolishness."

  First, we create a vocabulary by taking a list of unique words. Next, each sentence is converted to a vector where words are marked as present (1) or absent (0). For instance, using a predefined vocabulary of 10 words, the sentence "It was the best of times" can be represented as [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]. This approach enables consistent feature extraction across documents, making it scalable and effective for tasks like text classification. However, since BoW disregards word order and contextual meaning, it may struggle with capturing deeper semantic relationships [37].

- **Term Frequency-Inverse Document Frequency** : TF-IDF is a statistical technique used to determine the importance of a word in a document relative to a collection of documents. A word receives a **higher TF-IDF score** if it appears frequently in a document but is rare across the entire corpus, making it more relevant to that specific document. Conversely, a **lower TF-IDF score** indicates that a word is either common across many documents or appears infrequently in the given document.

  TF-IDF consists of two key components [76]:

  TF measures how often a word appears in a document. It is calculated as:

$$TF = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{A.1}$$

  IDF measures how rare or unique a word is across the entire corpus. It is calculated as:

$$IDF(t, D) = \log\left(\frac{\text{Total number of documents in corpus } D}{\text{Number of documents containing term } t}\right) \tag{A.2}$$

  The **TF-IDF score** for a term t in a document d is then computed as:

$$TF\text{-}IDF(t, d, D) = TF(t, d) \times IDF(t, D) \tag{A.3}$$

While TF-IDF is useful for identifying important words in documents, it has limitations. It does not capture context or the relationships between words, making it harder for the model to fully understand the text. Additionally, TF-IDF is sensitive to changes in document frequency, known as the extensive margin issue, where small variations in the dataset can significantly impact term importance scores, leading to inconsistencies. Another challenge is the zero value issue—when a word is missing from a document, it gets a zero weight, which can erase important contextual information. This can cause related words, like synonyms, to be treated as completely unrelated, reducing the model's ability to recognize meaningful connections in the text [15].

BIBLIOGRAPHY

[1] "(PDF) Addressing the Class Imbalance Problem in Medical Datasets". en. In: *ResearchGate* (). DOI: 10.7763/IJMLC.2013.V3.307. URL: https://www.researchgate.net/publication/239608168_Addressing_the_Class_Imbalance_Problem_in_Medical_Datasets (visited on 05/31/2025).

[2] *(PDF) Weight Initialization Techniques for Deep Learning Algorithms in Remote Sensing: Recent Trends and Future Perspectives*. en. Apr. 2025. DOI: 10.48550/arXiv.2102.07004. URL: https://www.researchgate.net/publication/349336188_Weight_Initialization_Techniques_for_Deep_Learning_Algorithms_in_Remote_Sensing_Recent_Trends_and_Future_Perspectives (visited on 06/11/2025).

[3] *[2106.06090] Graph Neural Networks for Natural Language Processing: A Survey*. URL: https://ar5iv.labs.arxiv.org/html/2106.06090 (visited on 05/12/2025).

[4] *A Comprehensive Guide to Attention Mechanism in Deep Learning for Everyone*. en. Nov. 2019. URL: https://www.analyticsvidhya.com/blog/2019/11/comprehensive-guide-attention-mechanism-deep-learning/ (visited on 03/03/2025).

[5] Jay Alammar. *The Illustrated Transformer*. URL: https://jalammar.github.io/illustrated-transformer/ (visited on 03/23/2025).

[6] Israa Alghanmi, Luis Espinosa Anke, and Steven Schockaert. "Combining BERT with Static Word Embeddings for Categorizing Social Media". en. In: *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*. Online: Association for Computational Linguistics, 2020, pp. 28–33. DOI: 10.18653/v1/2020.wnut-1.5. URL: https://www.aclweb.org/anthology/2020.wnut-1.5 (visited on 06/03/2025).

[7] Rawabi Aljadani and Mohammed Aseeri. "Prevalence of drug–drug interactions in geriatric patients at an ambulatory care pharmacy in a tertiary care teaching hospital". In: *BMC Research Notes* 11.1 (Apr. 2018), p. 234. ISSN: 1756-0500. DOI: 10.1186/s13104-018-3342-5. URL: https://doi.org/10.1186/s13104-018-3342-5 (visited on 02/23/2025).

[8] Hey Amit. *One Hot Encoding for Categorical Variables*. en. Nov. 2024. URL: https://medium.com/@heyamit10/one-hot-encoding-for-categorical-variables-095797f304c9 (visited on 03/22/2025).

[9] Ankiit. *Word2vec vs BERT*. en. Feb. 2024. URL: https://medium.com/@ankiit/word2vec-vs-bert-d04ab3ade4c9 (visited on 03/23/2025).

[10] Dogu Araci. *FinBERT: Financial Sentiment Analysis with Pre-trained Language Models*. arXiv:1908.10063 [cs]. Aug. 2019. DOI: 10.48550/arXiv.1908.10063. URL: http://arxiv.org/abs/1908.10063 (visited on 05/25/2025).

[11] Matthijs L Becker et al. "Hospitalisations and emergency department visits due to drug–drug interactions: a literature review". In: *Pharmacoepidemiology and drug safety* 16.6 (2007), pp. 641–651.

[12] Iz Beltagy, Kyle Lo, and Arman Cohan. *SciBERT: A Pretrained Language Model for Scientific Text*. arXiv:1903.10676 [cs]. Sept. 2019. DOI: 10.48550/arXiv.1903.10676. URL: http://arxiv.org/abs/1903.10676 (visited on 03/23/2025).

[13] *BERT Feature Extraction: A Comprehensive Guide*. URL: https://www.byteplus.com/en/topic/496892?title=bert-feature-extraction-unlocking-advanced-natural-language-processing-techniques (visited on 05/18/2025).

[14] Jari Bjorne et al. "Drug-Drug Interaction Extraction from Biomedical Texts with SVM and RLS Classifiers". en. In: ' ' ().

[15] *Challenges with TF-IDF in NLP: Key Insights*. URL: https://www.byteplus.com/en/topic/400333?title=challenges-with-tf-idf-in-nlp-key-insights (visited on 03/22/2025).

[16] *Deep learning for drug–drug interaction extraction from the literature: a review | Briefings in Bioinformatics | Oxford Academic*. URL: https://academic.oup.com/bib/article/21/5/1609/5610007 (visited on 02/25/2025).

[17] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv:1810.04805 [cs]. May 2019. DOI: 10.48550/arXiv.1810.04805. URL: http://arxiv.org/abs/1810.04805 (visited on 03/07/2025).

[18] Devendra Singh Dhami et al. "Drug-Drug Interaction Discovery: Kernel Learning from Heterogeneous Similarities". In: *Smart Health (Amst)* 9-10 (Dec. 2018), pp. 88–100. ISSN: 2352-6483. DOI: 10.1016/j.smhl.2018.07.007. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6289266/ (visited on 03/03/2025).

[19] Mingliang Dou et al. "Drug–Drug Interaction Relation Extraction Based on Deep Learning: A Review". In: *ACM Comput. Surv.* 56.6 (Mar. 2024), 158:1–158:33. ISSN: 0360-0300. DOI: 10.1145/3645089. URL: https://dl.acm.org/doi/10.1145/3645089 (visited on 03/02/2025).

[20] *Drug-Drug Interaction Extraction from Drug Labels*. URL: https://bionlp.nlm.nih.gov/tac2018druginteractions/ (visited on 03/07/2025).

[21] Mohsen Fatehifar and Hossein Karshenas. "Drug-Drug interaction extraction using a position and similarity fusion-based attention mechanism". In: *Journal of Biomedical Informatics* 115 (Mar. 2021), p. 103707. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2021.103707. URL: https://www.sciencedirect.com/science/article/pii/S1532046421000368 (visited on 05/23/2025).

[22] *Fine-tuning large language models (LLMs) in 2025 | SuperAnnotate*. URL: https://www.superannotate.com/blog/llm-fine-tuning%7D,%20urldate%20=%20%7B2025-05-25%7D,%20file%20=%20%7BFine-tuning%20large%20language%20models%20(LLMs)%20in%202025%20%7C%20SuperAnnotate:files/315/llm-fine-tuning.html:text/html%7D,.

[23] Hocine Gacem et al. "Les interactions médicamenteuses en clinique: étude prospective ciblant un service de cardiologie". In: *Batna J Med Sci* 1.1 (2014), pp. 2–6.

[24] Yu Gu et al. "Domain-Specific Language Model Pretraining for Biomedical Natural Language Processing". In: *ACM Trans. Comput. Healthcare* 3.1 (Jan. 2022). arXiv:2007.15779 [cs], pp. 1–23. ISSN: 2691-1957, 2637-8051. DOI: 10.1145/3458754. URL: http://arxiv.org/abs/2007.15779 (visited on 03/23/2025).

[25] David L. Hare et al. "Depression and cardiovascular disease: a clinical review". eng. In: *Eur Heart J* 35.21 (June 2014), pp. 1365–1372. ISSN: 1522-9645. DOI: 10.1093/eurheartj/eht462.

[26] María Herrero-Zazo et al. "The DDI corpus: An annotated corpus with pharmacological substances and drug–drug interactions". In: *Journal of Biomedical Informatics* 46.5 (Oct. 2013), pp. 914–920. ISSN: 1532-0464. DOI: 10.1016/j.jbi.2013.07.011. URL: https://www.sciencedirect.com/science/article/pii/S1532046413001123 (visited on 03/07/2025).

[27] Degen Huang et al. "Drug–drug interaction extraction from biomedical literature using support vector machine and long short term memory networks". In: *Information Sciences* 415-416 (Nov. 2017), pp. 100–109. ISSN: 0020-0255. DOI: 10.1016/j.ins.2017.06.021. URL: https://www.sciencedirect.com/science/article/pii/S002002551730110X (visited on 02/27/2025).

[28] Omar Hussein. *Omar Hussein*. en. URL: https://omar-hussein.medium.com (visited on 06/15/2025).

[29] Maryam KafiKang and Abdeltawab Hendawi. "Drug-Drug Interaction Extraction from Biomedical Text Using Relation BioBERT with BLSTM". en. In: *Machine Learning and Knowledge Extraction* 5.2 (June 2023). Number: 2 Publisher: Multidisciplinary Digital Publishing Institute, pp. 669–683. ISSN: 2504-4990. DOI: 10.3390/make5020036. URL: https://www.mdpi.com/2504-4990/5/2/36 (visited on 02/26/2025).

[30] Thomas N Kipf and Max Welling. "SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS". en. In: (2017).

[31] *Knowledge Mining of Interactions between Drugs from the Extensive Literature with a Novel Graph-Convolutional-Network-Based Method*. URL: https://www.mdpi.com/2079-9292/12/2/311 (visited on 02/25/2025).

[32] Mouna Labiadh. *Exploring BERT: Feature extraction & Fine-tuning*. en. Mar. 2024. URL: https://medium.com/dataness-ai/exploring-bert-feature-extraction-fine-tuning-6d6ad7b829e7 (visited on 05/18/2025).

[33] Jinhyuk Lee et al. "BioBERT: a pre-trained biomedical language representation model for biomedical text mining". In: *Bioinformatics* 36.4 (Feb. 2020), pp. 1234–1240. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz682. URL: https://doi.org/10.1093/bioinformatics/btz682 (visited on 03/23/2025).

[34] Sangrak Lim, Kyubum Lee, and Jaewoo Kang. "Drug drug interaction extraction from the literature using a recursive neural network". en. In: *PLOS ONE* 13.1 (Jan. 2018). Publisher: Public Library of Science, e0190926. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0190926. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0190926 (visited on 02/25/2025).

[35] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. arXiv:1708.02002 [cs]. Feb. 2018. DOI: 10.48550/arXiv.1708.02002. URL: http://arxiv.org/abs/1708.02002 (visited on 06/01/2025).

[36] Shengyu Liu et al. "Drug-Drug Interaction Extraction via Convolutional Neural Networks". en. In: *Computational and Mathematical Methods in Medicine* 2016.1 (2016). _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2016/6918381, p. 6918381. ISSN: 1748-6718. DOI: 10.1155/2016/6918381. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2016/6918381 (visited on 02/25/2025).

[37] *Machine Learning Mastery*. URL: machinelearningmastery (visited on 03/22/2025).

[38] Ben McMann. *The History of AI: From Rules-based Algorithms to Generative Models*. en-US. July 2024. URL: https://lanternstudios.com/insights/blog/the-history-of-ai-from-rules-based-algorithms-to-generative-models/ (visited on 02/24/2025).

[39] *Medical language – a unique linguistic phenomenonMedicinski jezik – jedinstveni lingvistički fenomen*. URL: https://www.researchgate.net/publication/334101480_Medical_language_-_a_unique_linguistic_phenomenonMedicinski_jezik_-_jedinstveni_lingvisticki_fenomen (visited on 02/25/2025).

[40] Ishani Mondal. "BERTKG-DDI: Towards Incorporating Entity-specific Knowledge Graph Information in Predicting Drug-Drug Interactions". en. In: ().

[41] Shalaka Naik. *What is Kernel in Machine Learning*. en-US. Jan. 2022. URL: https://www.educba.com/what-is-kernel-in-machine-learning/ (visited on 03/03/2025).

[42] *Named entity recognition*. en-US. URL: http://nlpprogress.com/english/named_entity_recognition.html (visited on 05/16/2025).

[43] *Natural Language Processing of Clinical Notes on Chronic Diseases: Systematic Review - PMC*. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC6528438/#sec9 (visited on 03/03/2025).

[44] Tuan Duc Ngo, Binh-Son Hua, and Khoi Nguyen. *ISBNet: a 3D Point Cloud Instance Segmentation Network with Instance-aware Sampling and Box-aware Dynamic Convolution*. arXiv:2303.00246 [cs]. Mar. 2023. DOI: 10.48550/arXiv.2303.00246. URL: http://arxiv.org/abs/2303.00246 (visited on 05/13/2025).

[45] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1310–1318. URL: https://proceedings.mlr.press/v28/pascanu13.html.

[46] "Prevalence and Factors Associated with Polypharmacy among Elderly Persons." en. In: *The Egypt. J. of Community Medicine* 37.3 (July 2019), pp. 55–61. ISSN: 2090-2611. DOI: 10.21608/ejcm.2019.43370. URL: https://ejcm.journals.ekb.eg/article_43370.html (visited on 02/24/2025).

[47] Alec Radford et al. *Improving Language Understanding by Generative Pre-Training.* en. Tech. rep. Technical Report. OpenAI, 2018. URL: https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

[48] Majid Rastegar-Mojarad, Richard D. Boyce, and Rashmi Prasad. "UWM-TRIADS: Classifying Drug-Drug Interactions with Two-Stage SVM and Post-Processing". In: *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013).* Ed. by Suresh Manandhar and Deniz Yuret. Atlanta, Georgia, USA: Association for Computational Linguistics, June 2013, pp. 667–674. URL: https://aclanthology.org/S13-2110/ (visited on 05/23/2025).

[49] Nils Reimers and Iryna Gurevych. "Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 338–348. DOI: 10.18653/v1/D17-1035. URL: https://aclanthology.org/D17-1035/ (visited on 06/03/2025).

[50] Anastasia Rivkin. "Admissions to a medical intensive care unit related to adverse drug reactions". In: *American Journal of Health-System Pharmacy* 64.17 (2007), pp. 1840–1843.

[51] Rahmat Sediq et al. "Concordance assessment of self-reported medication use in the Netherlands three-generation Lifelines Cohort study with the pharmacy database iaDB.nl: The PharmLines initiative". eng. In: *Clin Epidemiol* 10 (2018), pp. 981–989. ISSN: 1179-1349. DOI: 10.2147/CLEP.S163037.

[52] Isabel Segura-Bedmar, Paloma Martínez, and César de Pablo-Sánchez. "A linguistic rule-based approach to extract drug-drug interactions from pharmacological documents". In: *BMC Bioinformatics* 12.Suppl 2 (Mar. 2011), S1. ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-S2-S1. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3073181/ (visited on 02/26/2025).

[53] Leslie N. Smith. *Cyclical Learning Rates for Training Neural Networks.* arXiv:1506.01186 [cs]. Apr. 2017. DOI: 10.48550/arXiv.1506.01186. URL: http://arxiv.org/abs/1506.01186 (visited on 06/13/2025).

[54] Peng Su and K. Vijay-Shanker. "Investigation of improving the pre-training and fine-tuning of BERT model for biomedical relation extraction". In: *BMC Bioinformatics* 23.1 (Apr. 2022), p. 120. ISSN: 1471-2105. DOI: 10.1186/s12859-022-04642-w. URL: https://doi.org/10.1186/s12859-022-04642-w (visited on 06/02/2025).

[55] Xia Sun et al. “Drug-Drug Interaction Extraction via Recurrent Hybrid Convolutional Neural Networks with an Improved Focal Loss”. en. In: *Entropy* 21.1 (Jan. 2019). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 37. ISSN: 1099-4300. DOI: 10.3390/e21010037. URL: https://www.mdpi.com/1099-4300/21/1/37 (visited on 06/09/2025).

[56] *TAC DDI*. URL: https://bionlp.nlm.nih.gov/tac2019druginteractions/ (visited on 05/19/2025).

[57] Siliang Tang et al. *Two Step Joint Model for Drug Drug Interaction Extraction*. arXiv:2008.12704 [cs]. Aug. 2020. DOI: 10.48550/arXiv.2008.12704. URL: http://arxiv.org/abs/2008.12704 (visited on 02/26/2025).

[58] *The importance of NLP in biomedical research*. en. URL: https://blog.biostrand.ai/the-importance-of-nlp-in-biomedical-research (visited on 02/24/2025).

[59] *Think Topics | IBM*. URL: https://www.ibm.com/think/topics (visited on 05/09/2025).

[60] Robert Tinn et al. “Fine-tuning large neural language models for biomedical natural language processing”. en. In: *Patterns* 4.4 (Apr. 2023), p. 100729. ISSN: 26663899. DOI: 10.1016/j.patter.2023.100729. URL: https://linkinghub.elsevier.com/retrieve/pii/S2666389923000697 (visited on 03/23/2025).

[61] *Transformer Architecture: The Positional Encoding - Amirhossein Kazemnejad's Blog*. URL: https://kazemnejad.com/blog/transformer_architecture_positional_encoding (visited on 03/23/2025).

[62] *Transformer Models in Natural Language Processing*. en. URL: https://www.netguru.com/blog/transformer-models-in-nlp (visited on 03/23/2025).

[63] *Transformer Models: NLP's New Powerhouse*. en-US. URL: https://datasciencedojo.com/blog/transformer-models/ (visited on 05/17/2025).

[64] Esther Nkiruka Ugwu and Oluwatomi O Adeoti. “The Language of the Medical Profession: Doctor-Patient Discourse”. en. In: *LWATI: A Journal of Contemporary Research* 10.2 (2013), pp. 204–217. URL: https://www.ajol.info/index.php/lwati/article/view/92205.

[65] *Understanding and Coding the Self-Attention Mechanism of Large Language Models From Scratch*. URL: https://sebastianraschka.com/blog/2023/self-attention-from-scratch.html (visited on 03/06/2025).

[66] *Universal POS tags*. URL: https://universaldependencies.org/u/pos/ (visited on 02/21/2025).

[67] Ashish Vaswani et al. “Attention is All you Need”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 03/06/2025).

[68] Petar Veličković et al. “Graph Attention Networks”. en. In: *arXiv preprint arXiv:1710.10903* (2018). URL: https://arxiv.org/abs/1710.10903.

[69] Santiago Vilar, Carol Friedman, and George Hripcsak. "Detection of drug–drug interactions through data mining studies using clinical sources, scientific literature and social media". In: *Briefings in Bioinformatics* 19.5 (Sept. 2018), pp. 863–877. ISSN: 1477-4054. DOI: 10.1093/bib/bbx010. URL: https://doi.org/10.1093/bib/bbx010 (visited on 02/25/2025).

[70] Jason Wei and Kai Zou. *EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks*. arXiv:1901.11196 [cs]. Aug. 2019. DOI: 10.48550/arXiv.1901.11196. URL: http://arxiv.org/abs/1901.11196 (visited on 05/31/2025).

[71] *Welcome to the AI Glossary*. en. URL: https://deepgram.com/ai-glossary (visited on 06/14/2025).

[72] Yilin Wen, Zifeng Wang, and Jimeng Sun. *MindMap: Knowledge Graph Prompting Sparks Graph of Thoughts in Large Language Models*. arXiv:2308.09729 [cs]. Mar. 2024. DOI: 10.48550/arXiv.2308.09729. URL: http://arxiv.org/abs/2308.09729 (visited on 06/01/2025).

[73] *What are Graph Neural Networks (GNN) & How Do They Work?: A Comprehensive Guide*. URL: https://www.ionio.ai/blog/a-comprehensive-guide-about-graph-neural-networks-gnn (visited on 05/14/2025).

[74] *What Are Transformers in NLP: Benefits and Drawbacks*. en. URL: https://blog.pangeanic.com/what-are-transformers-in-nlp (visited on 03/03/2025).

[75] *What Is Dependency Parsing and How It Works - Wisdom ML*. en-US. Section: NLP Tutorial. Mar. 2023. URL: https://wisdomml.in/what-is-dependency-parsing-and-how-it-works/ (visited on 05/08/2025).

[76] *Word embeddings in NLP: A Complete Guide*. en. URL: https://www.turing.com/kb/guide-on-word-embeddings-in-nlp (visited on 03/22/2025).

[77] Wuti Xiong et al. "Extracting Drug-drug Interactions with a Dependency-based Graph Convolution Neural Network". In: *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2019, pp. 755–759. DOI: 10.1109/BIBM47256.2019.8983150.

[78] Dong Yuan, Frederic Maire, and Feras Dayoub. "Cross-Attention Between Satellite and Ground Views for Enhanced Fine-Grained Robot Geo-Localization". en. In: *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. Waikoloa, HI, USA: IEEE, Jan. 2024, pp. 1238–1245. ISBN: 979-8-3503-1892-0. DOI: 10.1109/WACV57701.2024.00128. URL: https://ieeexplore.ieee.org/document/10483962/ (visited on 06/02/2025).

[79] Jinrui Zhang. "How does the integration of syntactic features (POS tags or dependency parsing) during BERT fine-tuning influence the performance of semantic parsing?" en. In: (2025).

[80] Zhehuan Zhao et al. "Drug drug interaction extraction from biomedical literature using syntax convolutional neural network". In: *Bioinformatics* 32.22 (July 2016). _eprint: https://academic.oup.com/bioinformatics/article-pdf/32/22/3444/49027063/bioinformatics_32_22_3444.pdf, pp. 3444–3453. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btw486. URL: %5Curl%7Bhttps://doi.org/10.1093/bioinformatics/btw486%7D.

[81] Wei Zheng et al. “An attention-based effective neural model for drug-drug interactions extraction”. In: *BMC Bioinformatics* 18.1 (Oct. 2017), p. 445. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1855-x. URL: https://doi.org/10.1186/s12859-017-1855-x (visited on 02/26/2025).