Additional Materials for "Deciphering the Spectrum of Biomedical Knowledge through PubMed Data Mining"

1. Introduction

1.1 Project Background

This research aims to develop a comprehensive pipeline for analysing drug-target interactions through text mining of PubMed data. The approach in this research integrates advanced text mining techniques with network analysis to identify novel drug targets and investigate polypharmacological interactions.

1.2 Structure of the Additional Materials

The additional materials are designed to provide a detailed guide for reproducing our research. These materials are organised to follow the chronological flow of our research process:

Each section includes:

- The purpose of the step
- Input and output file names
- Relevant code snippets or examples

- Instructions for reproduction
- Expected outputs or visualisations

1.3 How to Use These Materials

To effectively use these materials for reproducing our research:

- Start by reviewing the “Computational Requirements” section to ensure you have the necessary hardware and software.
- Follow the “Reproduction Steps” in order, as each step often depends on the output of the previous steps.
- Pay close attention to file names and paths mentioned in each section to ensure you're using the correct input data.
- Refer to the code snippets and examples provided to understand the key processes and verify your outputs.
- Use the visualisations and output examples as benchmarks to compare your results.
- If you encounter any issues, check the "Additional Notes" section for troubleshooting tips.

All code snippets provided in this document are excerpts from the full scripts. Complete scripts can be found in the project's GitHub repository: https://github.com/jessicazhu06/Deciphering-the-Spectrum-of-Biomedical-Knowledge

2. Data Collection and Preparation

2.1 Data Collection

Script: ***1_data_collection.py***

Purpose:

- Automatically downloads the annual PubMed dataset, which is distributed across 1,219 individual XML files.
- Extract the following fields: ‘Date’, ‘Title’, ‘Abstract’, ‘Keyword’, ‘MeshHeadings’

Input: None (data is downloaded directly from the PubMed website)

Output: 1,219 XML files, each named in the format 'pubmed24n0001.xml', 'pubmed24n0002.xml', etc.

To reproduce:

- Ensure you have a stable internet connection and sufficient storage space (approximately 350GB).
- Run ***1_data_collection.py***

Packages used:

- pandas
- xml.etree.ElementTree

Note: *The download process may take several hours, depending on your internet speed. The script includes error handling and will attempt to resume downloads if interrupted.*

2.2 Data Preparation

Script: ***2_data_preparation.ipynb***

Purpose: This script processes the raw PubMed data, filtering for articles from 2019-2023.

Input: 1,219 XML files from PubMed (output of step 2.1)

Output: *sampled_pubmed_data.csv*

To reproduce:

1) Ensure all XML files from step 2.1 are in your designated input directory.
2) Open and run ***2_data_preparation.ipynb*** in a Jupyter notebook environment.
3) The script will:
    a) Load and parse the XML files
    b) Filter articles from 2019-2023
    c) Save the sampled data as ***sampled_pubmed_data.csv***

Packages used:

- pandas
- numpy

3. Fine-tuning BioBERT Model

Script: ***3_fine_tune_bio_bert.ipynb***

Purpose: This script fine-tunes the BioBERT model for Named Entity Recognition (NER) and Relationship Extraction (RE) tasks using the DrugProt dataset.

Input:

- *drugprot_training_abstracts.tsv*
- *drugprot_training_entities.tsv*
- *drugprot_training_relations.tsv*

Output:

- *trained_ner_model/* (directory containing the fine-tuned NER model)
- *trained_relation_model/* (directory containing the fine-tuned RE model)
- *tokenizer/* (directory containing the tokenizer)

To reproduce:

1) Download the DrugProt dataset from https://zenodo.org/records/5042151#.YNwojm7tbzA
2) Place the DrugProt files in the appropriate directory
3) Run ***3_fine_tune_bio_bert.ipynb***

Packages used:

- torch
- transformers
- pandas
- sklearn
- tqdm

Parameters for fine-tuning:

- Model: "dmis-lab/biobert-v1.1"
- Max sequence length: 256
- Batch size: 16
- Learning rate: 2e-5
- Number of epochs: 5

Example of entity annotation in DrugProt:

Article identifier (PMID)	Entity/ term number	Type of entity mention	Start character offset of the entity mention	End character offset of the entity mention	Text string of the entity mention
26646931	T1	CHEMICAL	475	484	terazosin
26646931	T2	GENE-Y	891	894	Bcl2

Example of relationship annotation in DrugProt:

Article identifier (PMID)	DrugProt Relation	Interactor argument 1	Interactor argument 2
26646931	INHIBITOR	Arg1:T1	Arg2:T2
12181427	PART-OF	Arg1:T3	Arg2:T22

4. Data Preprocessing, Cleaning, and Entity Recognition

Script: ***4_data_preprocessing_cleaning_NER_and_RE.ipynb***

Purpose: This script combines data preprocessing, cleaning, and NER using the fine-tuned BioBERT model.

Input: *sampled_pubmed_data.csv*

Output: *processed_pubmed_data_with_entities.csv*

4.1 Data Cleaning and Preprocessing

Process:

1) Index the dataset:
    a) Assign a unique 'article_id' to each row
    b) Set 'article_id' as the index of the DataFrame
2) Implement stratified sampling:
    a) Select 10% of the total articles to create a manageable subset while maintaining representation across years
3) Clean and preprocess the 'Title' and 'Abstract' fields:
    a) Remove specific punctuation marks ('[', ']', '?', '.')
    b) Remove HTML tags
    c) Standardise whitespace
    d) Expand abbreviations using spaCy's abbreviation detector
    e) Remove rows where 'Title' or 'Abstract' values are None

Packages used:

- pandas
- re (regular expressions)
- spacy
- scispacy

To reproduce:

1) Ensure all required packages are installed
2) Load the sampled data from *sampled_pubmed_data.csv*
3) Run the data cleaning and preprocessing steps in ***4_data_preprocessing_cleaning_NER_and_RE.ipynb***

4.2 Named Entity Recognition (NER)

Packages used:

- torch
- transformers

Example of NER output of 'Title':

```
{'title_entities': [('Terazosin', 'CHEMICAL'), ('Bcl2',
'GENE-Y')]}
```

To reproduce:

1) Run tokenization and NER steps in ***4_data_preprocessing_cleaning_NER_and_RE.ipynb***
2) The script will save the cleaned data with identified entities as *processed_pubmed_data_with_entities.csv*

By running this line

```
df_final_processed_data['Year'].value_counts().sort_index()
```

you can get yearly distribution of literature counts after data cleaning as following:

```
Year    Count
2019    112,128
2020    138,753
2021    147,977
2022    144,040
2023    148,994
```

Note: *since the file size limit in GitHub the process outputs of entity identification are not saved in the final output file.*

5. Relationship Extraction

Script: ***4_data_preprocessing_cleaning_NER_and_RE.ipynb*** (continued)

Purpose: This section of the script performs RE using the fine-tuned BioBERT model.

Input: *processed_pubmed_data_with_entities.csv*

Output: *pubmed_data_with_relationships.csv*

Packages used:

- torch
- transformers

Example of RE output:

```
{'article_id': '10', 'entity1': 'ion channel',
'entity1_type': 'GENE-Y','entity2': 'GPCR', 'GENE-N':
'Terazosin','relationship': 'AGONIST', 'confidence':
0.567602}
```

To reproduce:

- Load the processed data with entities from *processed_pubmed_data_with_entities.csv*

- Continue running ***4_data_preprocessing_cleaning_NER_and_RE.ipynb***
- The script will save the data with extracted relationships as *pubmed_data_with_relationships.csv*

Note: *since the file size limit in GitHub the process outputs of relationship extraction are not saved in the final output file. The relationship types predicted by the model include: INHIBITOR, DIRECT-REGULATOR, SUBSTRATE, ACTIVATOR, INDIRECT-UPREGULATOR, INDIRECT-DOWNREGULATOR, ANTAGONIST, PRODUCT-OF, PART-OF, AGONIST, AGONIST-ACTIVATOR, SUBSTRATE_PRODUCT-OF, and AGONIST-INHIBITOR.*

6. Exploratory Data Analysis (EDA)

Script: ***5_exploratory_data_analysis.ipynb***

Purpose: This script performs various analyses on the processed data to gain insights into the dataset's characteristics.

Input:

- *processed_pubmed_data_with_entities.csv*
- *pubmed_data_with_relationships.csv*

Output: Various plots and statistics of the input datasets

6.1 Word Cloud for Entities

Key function: ***generate_wordclouds()***, ***analyze_entities()***

Outputs: Word cloud of abstracts '*Word Cloud of GENE-Y Entities.png'*

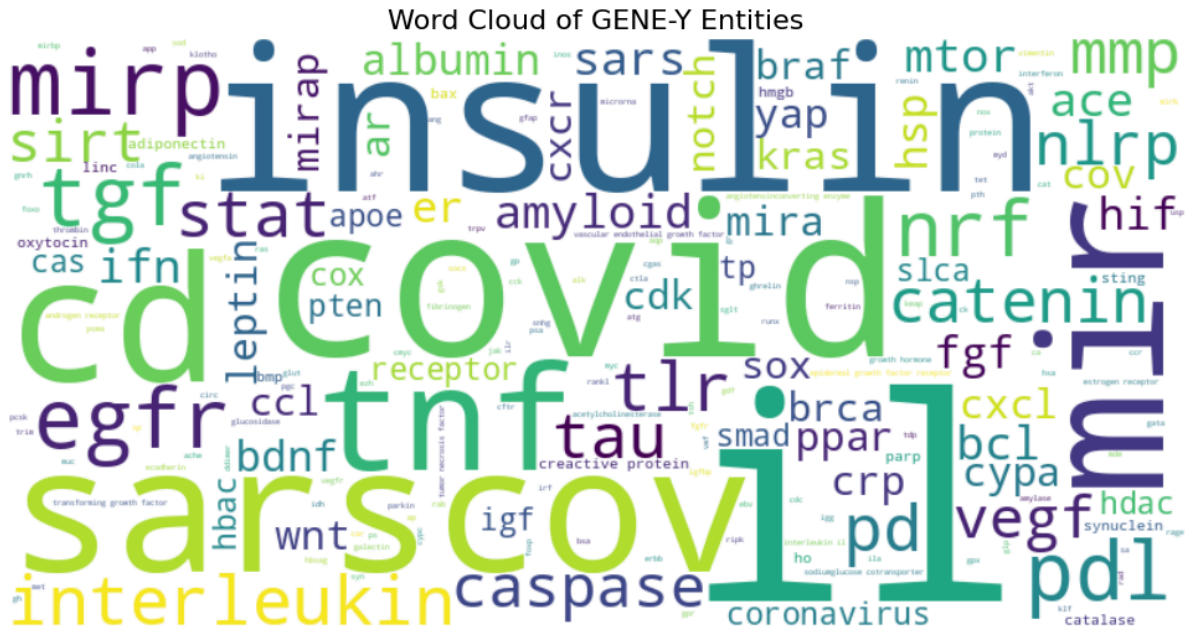


Figure 6.1: Word Cloud of GENE-N Entities (Example Output)

The word clouds visualise the most frequent terms in GENE-Y entities, offering a quick overview of the most common topics in the dataset.

6.2 Relationship Analysis

Key function: ***analyze_relationships()***

Output: Bar plot showing relationship distributions 'relationship_distribution.png'

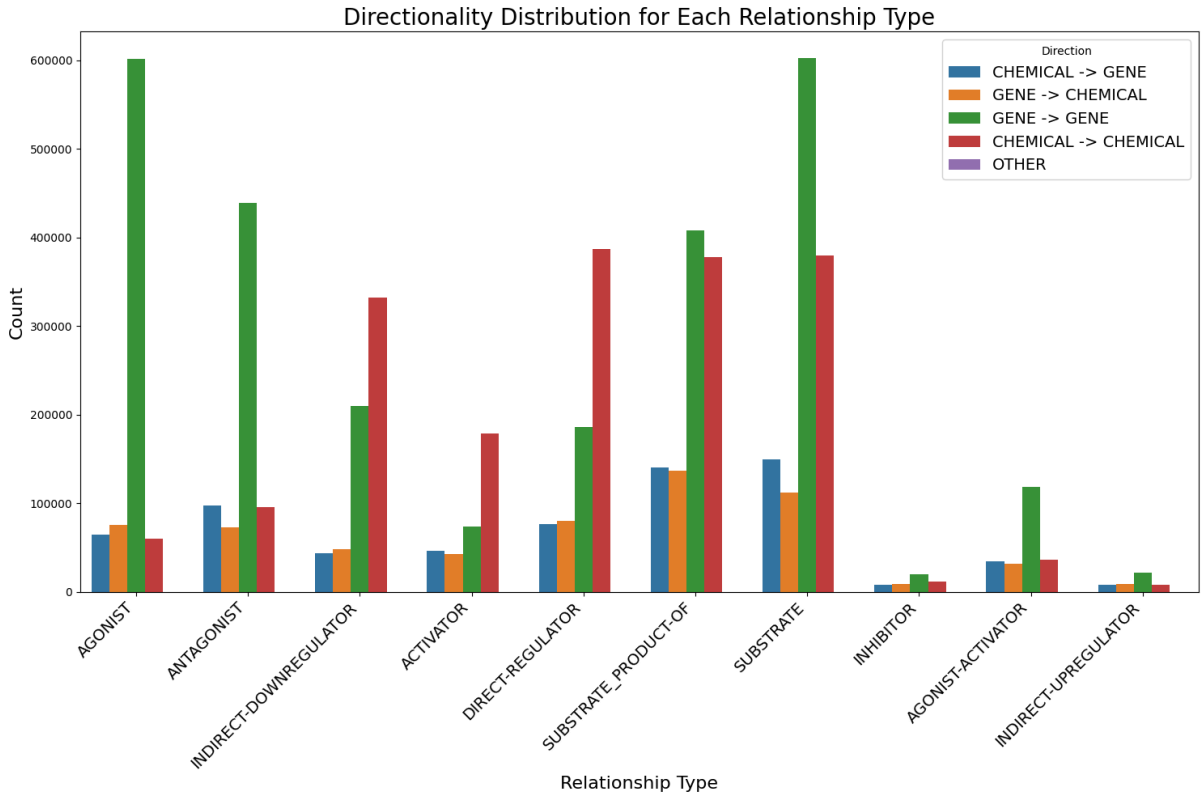


Figure 6.2: Directionality Distribution of Each Relationship Type (Example Output)

This plot illustrates the distribution of different types of relationships between entities, providing insights into the most common interactions in the dataset.

Packages used:

- pandas
- matplotlib
- seaborn
- wordcloud
- nltk

To reproduce:

1) Load the data with relationships from
   processed_pubmed_data_with_entities.csv
   pubmed_data_with_relationships.csv

2) Run ***5_exploratory_data_analysis.ipynb***
3) The script will generate and save all the plots mentioned above

7. Network Construction and Analysis

Script:

6_novel_targets_identification_and_polypharmacology_analysis_(network_construction).ipynb

Purpose: This script constructs a network from the extracted relationships and performs analysis to identify novel targets and polypharmacological interactions.

Input: *pubmed_data_with_relationships.csv*

Output:

- *drug_target_network.gpickle (NetworkX graph file)*
- *novel_targets.csv*
- *polypharmacology_candidates.csv*

7.1 Network Construction

This section constructs a graph from the extracted relationships.

```
def create_graph_from_relationships(df):
    G = nx.Graph()
    for _, row in tqdm(df.iterrows(), total=len(df)):
        entity1, entity2 = row['entity1'], row['entity2']
        entity1_type, entity2_type = row['entity1_type'],
row['entity2_type']
        relationship = row['relationship']
        confidence = row['confidence']
        article_id = str(row['article_id'])

        base_weight = 2.0 if ((entity1_type == 'CHEMICAL' and
entity2_type in ['GENE-Y', 'GENE-N']) or
```

```
                                (entity2_type == 'CHEMICAL' and
entity1_type in ['GENE-Y', 'GENE-N'])) else 1.0

        G.add_edge(entity1, entity2, base_weight=base_weight,
confidence=confidence,
                   relationship=relationship,
weight=base_weight * confidence,
                   article_id=[article_id])

        G.nodes[entity1]['type'] = entity1_type
        G.nodes[entity2]['type'] = entity2_type

    return G
```

7.2 Novel Target Identification

This section identifies potential novel targets based on network properties.

```
def identify_novel_targets(G, degree_cent, pagerank,
confidence_threshold=0.70, percentile_threshold=90):
    novel_targets = []
    for node, data in G.nodes(data=True):
        if data['type'] in ['GENE-Y', 'GENE-N']:
            degree = degree_cent[node]
            pr = pagerank[node]
            node_confidences =
[G[node][neighbor]['confidence'] for neighbor in
G.neighbors(node)]
            avg_confidence = np.mean(node_confidences) if
node_confidences else 0
            novelty_score = 2 if data['type'] == 'GENE-N'
else 1

            if (pr > np.percentile(list(pagerank.values()),
percentile_threshold) and
                degree <
np.percentile(list(degree_cent.values()),
percentile_threshold) and
                avg_confidence > confidence_threshold):
```

```
                combined_score = (pr * novelty_score) /
degree if degree != 0 else 0
                novel_targets.append((node, data['type'],
degree, pr, avg_confidence, combined_score))

    return pd.DataFrame(novel_targets, columns=['Target',
'Type', 'Degree', 'PageRank', 'AvgConfidence',
'CombinedScore'])
```

7.3 Polypharmacology Analysis

This section identifies potential polypharmacological interactions.

```
def analyze_polypharmacology(G, degree_cent, pagerank,
confidence_threshold=0.7, top_n=100):
    poly_candidates = []
    for node, data in G.nodes(data=True):
        if data['type'] == 'CHEMICAL':
            degree = degree_cent[node]
            pr = pagerank[node]
            targets = list(G.neighbors(node))
            num_targets = len(targets)

            if num_targets > 1:
                avg_confidence =
np.mean([G[node][target]['confidence'] for target in
targets])
                target_types = [G.nodes[target]['type'] for
target in targets]
                gene_y_count = sum(1 for t in target_types if
t == 'GENE-Y')
                gene_n_count = sum(1 for t in target_types if
t == 'GENE-N')
                poly_score = ((gene_y_count + 1.5 *
gene_n_count) * pr) / degree if degree != 0 else 0

                if avg_confidence > confidence_threshold:
                    poly_candidates.append((node,
```

```
num_targets, gene_y_count, gene_n_count, degree, pr,
avg_confidence, poly_score))

    return pd.DataFrame(poly_candidates, columns=['Drug',
'NumTargets', 'GENE-Y_Targets', 'GENE-N_Targets', 'Degree',
'PageRank', 'AvgConfidence', 'PolyScore'])
```

Packages used:

- networkx
- pandas
- numpy
- matplotlib
- seaborn

To reproduce:

1) Load the data with relationships from *pubmed_data_with_relationships.csv*
2) Run ***6_novel_targets_identification_and_polypharmacology_analysis_(network_construction).ipynb***
3) The script will generate and save:
    a) *The network as drug_target_network.gpickle*
    b) *Novel targets as novel_targets.csv*
    c) *Polypharmacology candidates as polypharmacology_candidates.csv*

7.4 Subgraph Generation for Novel Targets and Polypharmacology Interactions

Purpose: Generate and visualise subgraphs for identified novel targets and polypharmacology candidates.

Input: ***drug_target_network.gpickle***

Output: *Subgraph visualisations (PNG files)*

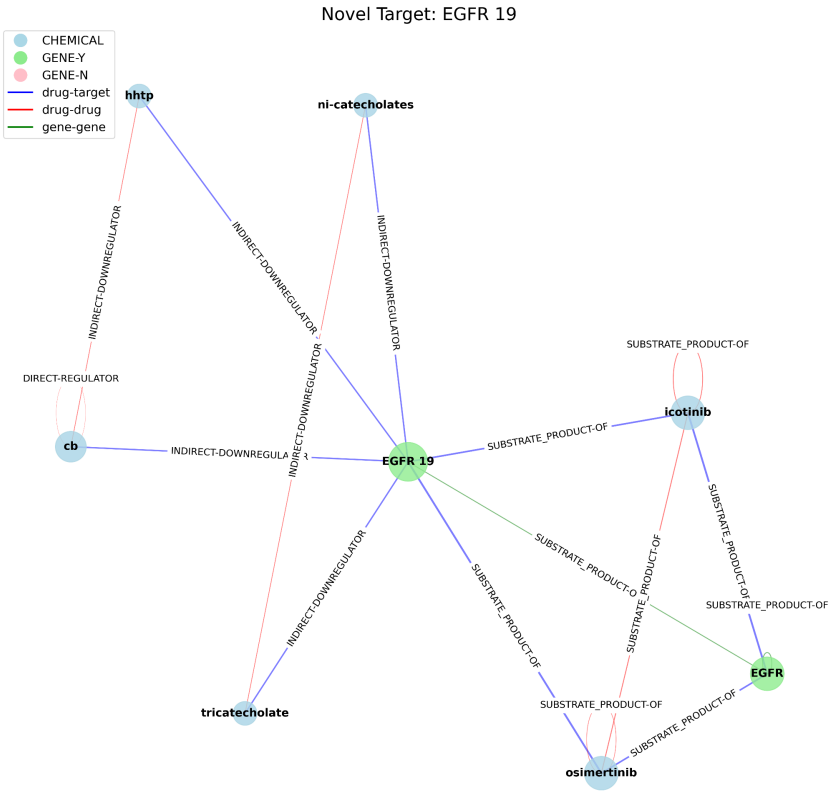
Key functions:

- *plot_novel_target_subgraph():* Generates subgraph for a novel target

- *plot_polypharmacology_subgraph()*: Generates subgraph for a polypharmacology candidate

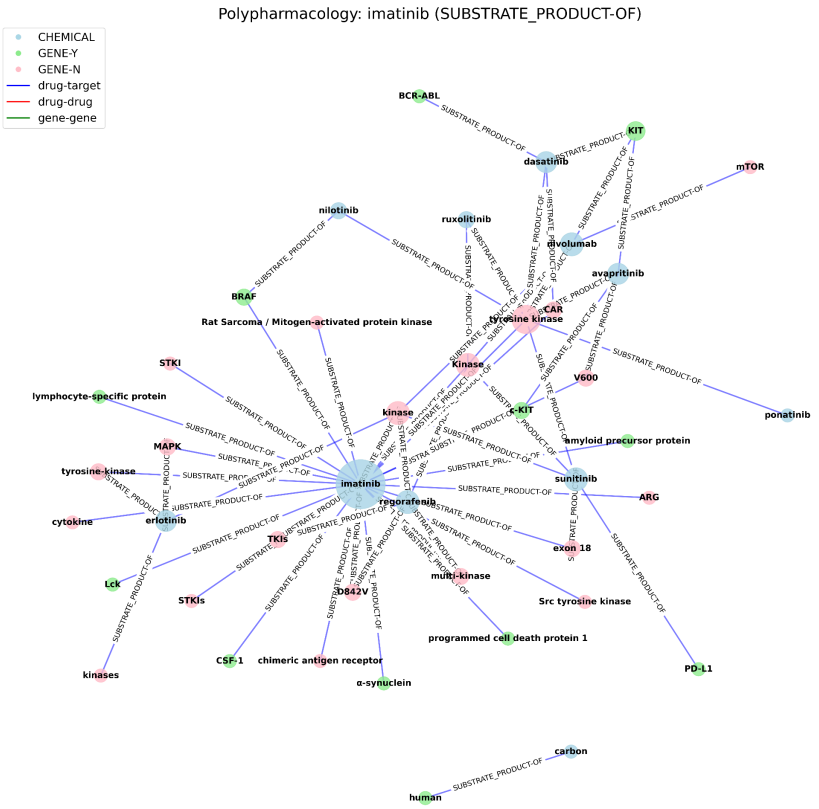
Example outputs:

1) Novel Target Subgraph:



This figure shows the local network structure around the novel target EGFR19, highlighting its interactions with various chemicals and other genes.

2) Polypharmacology Candidate Subgraph:



This figure illustrates the multiple targets of the drug imatinib, demonstrating its polypharmacological properties.

7.5 Comprehensive Analysis of Novel Targets and Polypharmacological Interactions

Purpose: Provide visualisations and statistical analysis of the identified novel targets and polypharmacology candidates.

Input:

- novel_targets.csv
- polypharmacology_candidates.csv

Output: Various plots and statistical summaries

Key functions:

- ***plot_metric_distributions()***: Visualises the distribution of network metrics
- ***compare_gene_types()***: Compares GENE-Y and GENE-N entities
- ***analyze_top_candidates()***: Analyses top novel targets and polypharmacology candidates

To reproduce:

1) Ensure you have the network graph file ***drug_target_network.gpickle*** and the CSV files for novel targets and polypharmacology candidates.
2) Run the script ***6_novel_targets_identification_and_polypharmacology_analysis_(network_construction).ipynb***
3) The resulting plots will be saved as PNG files, and statistical summaries will be printed to the console.

Note: *The specific novel targets and polypharmacology candidates chosen for subgraph visualisation are based on their Combined Score or PolyScore, respectively.*

8. Validation

Script: ***7_validation.ipynb***

Purpose: This script prepares a validation dataset and evaluates the performance of different models for entity recognition and relationship extraction tasks.

8.1 Validation Dataset Preparation

Process:

- Load the DrugProt dataset files:
    - drugprot_training_abstracts.tsv
    - drugprot_training_entities.tsv
    - drugprot_training_relations.tsv
- Sample 200 unique article IDs from the validation sets
- Filter entities and relations based on sampled PMIDs
- Extract abstracts for sampled PMIDs

Output:

- *sampled_entities_from_validation.csv*
- *sampled_relations_from_validation.csv*
- *sampled_abstracts_from_validation.csv*

8.2 Named Entity Recognition (NER) Validation

Models evaluated:

- Dictionary-based approach
- Pre-trained BioBERT
- Fine-tuned BioBERT

Process:

- Apply each model to the validation dataset
- Generate entity predictions for each model

Output files:

- *dictionary_based_results.csv*
- *biobert_results.csv*
- *fine_tuned_biobert_results.csv*

Comparison method:

- Performance metrics (precision, recall, F1-score) for entity recognition and classification

```
def evaluate_entity_recognition(ground_truth, model_results):
    # ... (implementation details)
    return precision, recall, f1_score

# Evaluate each model
for model_name, results_file in [("Dictionary-based",
"dictionary_based_results.csv"),
                                 ("BioBERT",
"biobert_results.csv"),
                                 ("Fine-tuned BioBERT",
"fine_tuned_biobert_results.csv")]:
    precision, recall, f1 =
evaluate_entity_recognition(ground_truth,
load_results(results_file))
    print(f"{model_name} - Precision: {precision:.4f},
Recall: {recall:.4f}, F1: {f1:.4f}")
```

- Comparison output:

```
Entity Recognition Performance:
Dictionary-based:
  Precision: 0.4922
  Recall: 0.2733
  F1-score: 0.3514

Fine-tuned BioBERT:
  Precision: 0.7202
  Recall: 0.7556
  F1-score: 0.7375

BioBERT:
  Precision: 0.0501
  Recall: 0.3495
  F1-score: 0.0876
```

8.3 Relationship Extraction (RE) Validation

Models evaluated:

- Machine Learning-based approach
- Pre-trained BioBERT
- Fine-tuned BioBERT

Process:

- Apply each model to the validation dataset
- Generate relationship predictions for each model

Output files:

- *ml_based_pairs.csv*
- *biobert_pairs.csv*
- *fine_tuned_biobert_pairs.csv*

Comparison method:

- Score distribution plots for relationship extraction methods

```
def plot_metric_distributions(rule_based, ml_based,
fine_tuned):
```

```
    plt.figure(figsize=(15, 5))
    # ... (implementation details)

plt.savefig('relationship_extraction_score_distributions.png'
)
    plt.close()

# Load results and plot distributions
ml_df = pd.read_csv('ml_based_pairs.csv')
biobert_df = pd.read_csv('biobert_pairs.csv')
fine_tuned_biobert_df =
pd.read_csv('fine_tuned_biobert_pairs.csv')
plot_metric_distributions(ml_df, biobert_df,
fine_tuned_biobert_df)
```

- Output of score distribution plots:

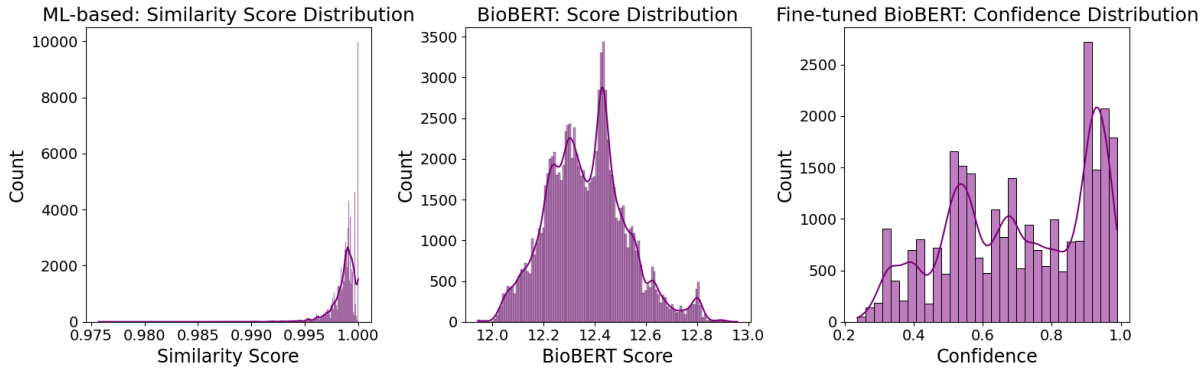


Figure 8.1: Score Distribution Comparison of Relationship Extraction Methods

To reproduce:

1) Ensure you have the DrugProt dataset files and the output files from NER and RE tasks.
2) Run ***7_validation.ipynb.***
3) Review the console output for NER performance metrics.
4) Check the generated plot *relationship_extraction_score_distributions.png* for RE score distributions.

9. Computational Requirements

To reproduce this research, you will need:

- Python 3.7+
- NVIDIA GPU with at least 8GB of VRAM (for model training)
- CUDA-enabled GPU (CUDA 10.1 or higher)

Key Python packages:

- torch
- transformers
- pandas
- numpy
- networkx
- matplotlib
- seaborn
- scikit-learn
- nltk
- spacy (with en_core_sci_sm model)
- scispacy

A complete list of required packages and their versions can be found in the ***requirements.txt*** file in the project repository.

10. Reproduction Steps

To reproduce this research:

1) Clone the project repository:

```
git clone [your-repo-url]
```

Install required packages:

```
pip install -r requirements.txt
```

2) Download the necessary datasets:
    a) PubMed data (2019-2023) from https://pubmed.ncbi.nlm.nih.gov/download/
    b) DrugProt dataset from https://zenodo.org/records/5042151#.YNwojm7tbzA
3) Run the scripts in the following order:

1_data_collection.py

2_data_preparation.ipynb

3_fine_tune_bio_bert.ipynb

4_data_preprocessing_cleaning_NER_and_RE.ipynb

5_exploratory_data_analysis.ipynb

6_novel_targets_identification_and_polypharmacology_analysis_(network_construction).ipynb

7_validation.ipynb

4) Compare your outputs with the expected outputs provided in each section of this document.

***Note**: Some processes, particularly NER and RE, may take several hours to complete depending on your hardware.*

11. Additional Notes

- File paths: In Colab, we use '/content/drive/MyDrive/' to access Google Drive. On a local machine, you'll need to adjust these paths.
- The plots and visualisations mentioned in the EDA and Network Analysis sections will be generated automatically when running the respective scripts. Make sure to check the output directory for these files.
- When fine-tuning the BioBERT model, you may need to adjust the batch size or use gradient accumulation if you encounter memory issues on your GPU.
- The novel target identification and polypharmacology analysis processes involve some randomness due to the nature of network analysis algorithms. While the overall trends should be consistent, exact numbers may vary slightly between runs.