VKGPharma: A Neural Pipeline for Vietnamese Pharmaceutical Knowledge Graph Question Answering using Natural Language to Cypher Translation

Huynh Thien Thuan (22127407)
Khuu Thanh Thien (22127396)
Bui Le Khoi (22127205)
Tran So Vinh (23127282)

University of Information Technology
Ho Chi Minh City, Vietnam
htthuan22@clc.fitus.edu.vn, ktthien22@clc.fitus.edu.vn
blkhoi22@clc.fitus.edu.vn, tsvinh23@clc.fitus.edu.vn

December 2025

Abstract

Knowledge graphs (KGs) have emerged as powerful tools for representing structured domain knowledge, enabling intelligent question answering systems. However, querying KGs typically requires expertise in formal query languages like Cypher, posing a significant barrier for end users. In this work, we present a comprehensive neural pipeline for Vietnamese pharmaceutical knowledge graph question answering that translates natural language queries (NLQ) directly to executable Cypher queries. Our system constructs a domain-specific knowledge graph from Vietnamese pharmaceutical documents using large language model-based relation extraction, then fine-tunes a BartPho sequence-to-sequence model on 5,414 synthetically generated question-Cypher pairs. We incorporate entity linking using multilingual E5 embeddings and query refinement with Gemini Flash to improve robustness. Experimental results demonstrate that our approach achieves 65.99% hard exact match accuracy and 73.01% soft exact match accuracy on the test set, showing promising performance for low-resource Vietnamese NLP tasks. The system is deployed as an interactive Streamlit application, providing accessible pharmaceutical knowledge retrieval for Vietnamese-speaking users. Our work contributes a complete end-to-end pipeline, a novel Vietnamese pharmaceutical knowledge graph, and a synthetic NLQ-to-Cypher dataset for future research.

Keywords: Knowledge Graph, Question Answering, Vietnamese NLP, Natural Language to Cypher, BartPho, Entity Linking, Pharmaceutical Domain

1 Introduction

1.1 Motivation

The pharmaceutical domain contains vast amounts of critical information about drugs, chemicals, diseases, and their intricate relationships. In Vietnam, pharmaceutical knowledge is predominantly documented in Vietnamese, creating a significant barrier for automated information retrieval systems that could benefit healthcare professionals, researchers, and patients. Traditional keyword-based search methods fail to capture the semantic relationships and complex queries that domain experts require, such as "What testing methods are required for drugs that

treat headaches?" or "Which storage conditions are needed for medications containing specific chemical compounds?"

Knowledge graphs (KGs) have emerged as a powerful paradigm for representing structured domain knowledge, where entities (drugs, diseases, chemicals) are nodes and relationships (treats, contains, tested_by) are edges [1]. However, querying knowledge graphs typically requires expertise in formal query languages such as Cypher (for Neo4j) or SPARQL (for RDF stores), which presents a significant usability barrier for non-technical users. This challenge is particularly acute in low-resource language settings like Vietnamese, where sophisticated natural language understanding tools are limited.

1.2 Problem Statement

This work addresses the following research question: *How can we enable Vietnamese-speaking users to query a pharmaceutical knowledge graph using natural language, without requiring knowledge of formal query languages?*

The problem encompasses several technical challenges:

- **Knowledge extraction**: Automatically extracting structured pharmaceutical knowledge from unstructured Vietnamese text documents
- **Low-resource NLP**: Developing effective natural language understanding models for Vietnamese, a language with limited annotated training data
- **Semantic parsing**: Translating natural language questions to executable Cypher queries while preserving semantic intent
- **Entity disambiguation**: Resolving entity mentions in queries to their corresponding knowledge graph identifiers
- **Robustness**: Handling morphological variations, spelling errors, and paraphrases common in natural language

1.3 Contributions

We present a complete end-to-end neural pipeline for Vietnamese pharmaceutical knowledge graph question answering. Our main contributions are:

1. **Vietnamese Pharmaceutical Knowledge Graph**: We construct a domain-specific knowledge graph from Vietnamese pharmaceutical documentation, containing 8 entity types (Drug, Chemical, Disease, Organism, Test Method, Standard, Storage Condition, Production Method) and 8 relationship types, extracted using large language model-based relation extraction.
2. **Synthetic NLQ-to-Cypher Dataset**: We create a dataset of 5,414 Vietnamese natural language question-Cypher query pairs using a systematic generation framework, addressing the scarcity of annotated Vietnamese semantic parsing data.
3. **Neural Translation Pipeline**: We develop a multi-stage pipeline that combines:
    - Fine-tuned BartPho [2] for draft Cypher generation
    - Gemini Flash for syntax refinement
    - Multilingual E5 embeddings [3] for entity linking
    - Neo4j for query execution

4. **Empirical Evaluation**: We demonstrate that our approach achieves 65.99% hard exact match accuracy and 73.01% soft exact match (variable-normalized) accuracy on held-out test questions, showing promising performance for a low-resource language semantic parsing task.

5. **Deployed System**: We release an interactive Streamlit web application that enables Vietnamese-speaking users to query pharmaceutical knowledge using natural language, with full source code and trained models available for reproducibility.

1.4 Paper Organization

The remainder of this paper is organized as follows: Section 2 surveys related work in knowledge graph question answering, Vietnamese NLP, and semantic parsing. Section 3 describes our data extraction process and synthetic question generation methodology. Section 4 details our neural pipeline architecture. Section 5 presents our experimental setup and training procedure. Section 6 reports quantitative results and qualitative analysis. Section 7 discusses limitations and future directions. Finally, Section 8 concludes the paper.

2 Related Work

2.1 Knowledge Graph Question Answering

Knowledge graph question answering (KGQA) has been extensively studied in recent years, with approaches broadly categorized into semantic parsing-based and information retrieval-based methods [4]. Semantic parsing approaches translate natural language questions into structured queries (e.g., SPARQL, Cypher) that can be executed against the knowledge graph [5, 6]. Early work relied on rule-based grammars and lexicons [7], but recent advances leverage neural sequence-to-sequence models for end-to-end learning [8, 9].

For graph databases specifically, several works have explored natural language to Cypher translation. [10] introduced a sketch-based approach for SQL generation that influenced later Cypher work. [11] surveyed neural approaches to graph query language generation, noting challenges in handling complex nested queries and aggregations. More recently, large language models have shown promise for few-shot semantic parsing [12, 13], though they still struggle with domain-specific schemas and low-resource languages.

2.2 Vietnamese Natural Language Processing

Vietnamese poses unique challenges for NLP due to its monosyllabic nature, extensive use of diacritics, and lack of explicit word boundaries [14]. Early Vietnamese NLP research focused on fundamental tasks such as word segmentation [15], part-of-speech tagging [16], and named entity recognition [17].

The emergence of pre-trained language models has significantly advanced Vietnamese NLP. PhoBERT [18] adapted the RoBERTa architecture for Vietnamese using a 20GB corpus, achieving state-of-the-art results on multiple benchmarks. For sequence-to-sequence tasks, VietAI released ViT5 [19], a Vietnamese adaptation of T5, and BartPho [2], based on the BART architecture. These models have been successfully applied to machine translation [20], text summarization [19], and abstractive question answering [21].

However, Vietnamese semantic parsing remains understudied. Existing work on Vietnamese question answering primarily focuses on reading comprehension [21] rather than structured knowledge base querying. Our work addresses this gap by developing the first Vietnamese natural language to Cypher translation system for knowledge graph question answering.

2.3 Semantic Parsing for Low-Resource Languages

Training semantic parsers typically requires large annotated datasets mapping natural language to formal queries, which are expensive to create and scarce for low-resource languages. Several strategies have been proposed to address this challenge:

Synthetic data generation creates training pairs programmatically. [22] generated synthetic SQL queries and used templates to create corresponding questions. [23] introduced a more sophisticated grammar-based approach for cross-domain SQL generation. Recent work by [24] developed the RNG-KBQA framework for synthetic SPARQL question generation over knowledge bases.

Cross-lingual transfer leverages resources from high-resource languages. [25] showed that translating English training data to target languages can improve multilingual semantic parsing. [26] used iterative back-translation to create training data for low-resource SQL parsing.

Few-shot learning with large language models has emerged as a promising alternative. [12] demonstrated that GPT-3 can perform competitive semantic parsing with minimal examples. [27] showed that constrained decoding with language models enables few-shot semantic parsing.

Our approach combines synthetic data generation with fine-tuning of Vietnamese pre-trained models, adapting techniques from high-resource languages to the Vietnamese pharmaceutical domain.

2.4 Domain-Specific Knowledge Graphs

Domain-specific knowledge graphs have been constructed for various specialized fields. In healthcare, the Unified Medical Language System (UMLS) [28] integrates biomedical vocabularies, while DrugBank [29] provides comprehensive drug information. For biomedical knowledge graphs, PharmKG [30] established a comprehensive benchmark for pharmaceutical data mining, and Hetionet [31] systematically integrated biomedical knowledge for drug repurposing.

Automatic knowledge graph construction from unstructured text has been extensively studied [1]. Traditional approaches used named entity recognition and relation extraction pipelines [32]. Recent work leverages pre-trained language models for joint entity and relation extraction [33] or employs large language models for zero-shot knowledge extraction [34].

To our knowledge, this is the first work to construct a Vietnamese pharmaceutical knowledge graph with an associated natural language query interface, contributing both the knowledge resource and the neural translation system.

3 Dataset

Our dataset consists of two main components: (1) a Vietnamese pharmaceutical knowledge graph constructed from extracted documents, and (2) a synthetic dataset of natural language question-Cypher query pairs for training the translation model. This section describes the data acquisition, preprocessing, knowledge graph construction, and synthetic question generation processes.

3.1 Source Documents

3.1.1 Document Acquisition

The source material consists of Vietnamese pharmaceutical documentation obtained from scanned PDF files. These documents contain comprehensive information about drug monographs, chemical compositions, testing procedures, storage requirements, and therapeutic applications following Vietnamese pharmaceutical standards.

3.1.2 Text Extraction

We developed a multi-stage text extraction pipeline implemented in `notebooks/extract_text.ipynb` to process the scanned PDFs:

1. **PDF Parsing**: We used PDFPlumber and PyPDF2 libraries to extract raw text from PDF pages, handling both native text and OCR-processed scanned images.
2. **Text Normalization**: The extracted text underwent extensive cleaning to address OCR artifacts, inconsistent spacing, and formatting irregularities common in scanned documents. We applied:
   - Diacritic normalization for Vietnamese characters
   - Whitespace standardization
   - Removal of page headers, footers, and page numbers
   - Correction of common OCR errors (e.g., "đ" misread as "d")
3. **Document Segmentation**: The cleaned text was segmented into coherent chunks representing distinct pharmaceutical entities or relationships, stored in `data/raw/text_chunks.json`.
4. **LLM-based Normalization**: We employed Google Gemini to further normalize the extracted text, correcting domain-specific terminology and ensuring consistency in pharmaceutical nomenclature (implemented in `notebooks/nomalization-with-LLM.ipynb`).

The final processed text corpus comprises pharmaceutical information spanning multiple drug categories, chemical compounds, diseases, and testing methodologies.

3.2 Knowledge Graph Construction

3.2.1 Schema Design

We designed a domain-specific schema capturing the essential entities and relationships in pharmaceutical knowledge:

Entity Types (8 node labels):

- **DRUG**: Pharmaceutical products, vaccines, and biological preparations
- **CHEMICAL**: Chemical compounds, active ingredients, excipients, and reagents
- **DISEASE**: Medical conditions and clinical symptoms
- **ORGANISM**: Bacteria, viruses, and cell lines
- **TEST_METHOD**: Quality control and analytical techniques
- **STANDARD**: Technical specifications (pH, concentration, potency)
- **STORAGE_CONDITION**: Environmental storage requirements
- **PRODUCTION_METHOD**: Manufacturing and formulation processes

Relationship Types (8 edge labels):

- **TREATS**: Drug/chemical treats disease
- **CONTAINS**: Drug contains chemical component
- **TARGETS**: Active ingredient acts on organism/organ

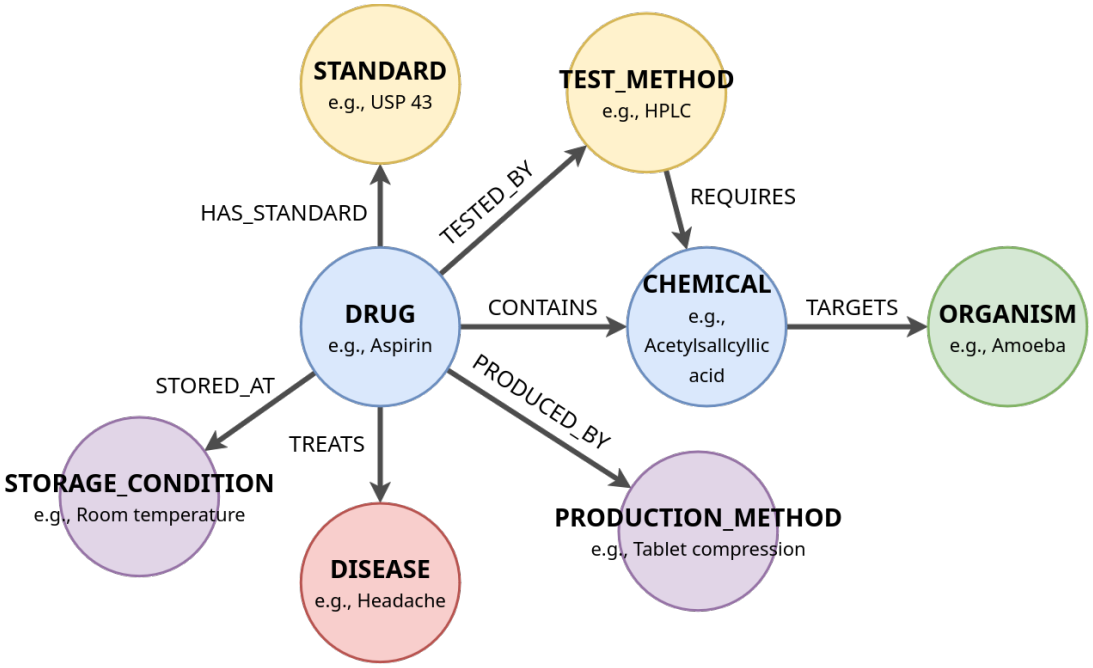


Figure 1: Pharmaceutical knowledge graph schema with entity and relationship types.

- **HAS_STANDARD**: Entity has technical specification
- **TESTED_BY**: Drug tested by analytical method
- **REQUIRES**: Method requires chemical/reagent/equipment
- **PRODUCED_BY**: Drug produced by method/organism
- **STORED_AT**: Drug stored at environmental condition

3.2.2 Relation Extraction

We employed a large language model-based approach for relation extraction from the normalized text chunks:

1. **Prompt Engineering**: We designed structured prompts that instruct the LLM to identify entities and relationships according to our schema. The prompts included:
   - Schema definitions with examples
   - Instructions for entity boundary detection
   - Guidelines for relationship inference
   - Output format specification (JSON)
2. **Batch Processing**: Text chunks were processed in batches using Google Gemini Flash, balancing throughput and cost. Each chunk yielded a set of entities and relationships.
3. **Entity Deduplication**: Extracted entities were normalized and deduplicated based on string similarity and domain knowledge. For example, "Paracetamol", "paracetamol", and "Para-cetamol" were merged into a single entity.
4. **Graph Population**: The extracted triples were loaded into a Neo4j graph database (implemented in `notebooks/build_KG.ipynb`) using the following process:

- Nodes were created with `MERGE` operations to ensure uniqueness
- Properties (entity type, raw text) were attached to nodes
- Relationships were established between matched node pairs

The resulting knowledge graph is stored in `data/neo4j/` and contains thousands of entities and relationships representing Vietnamese pharmaceutical knowledge.

3.3 Synthetic Question-Cypher Dataset Generation

Given the absence of annotated Vietnamese question-Cypher pairs, we developed a systematic framework for synthetic data generation, adapting techniques from [22, 24].

3.3.1 Generation Framework

We employed the **easy-dataset** framework with carefully designed prompts to generate diverse natural language questions paired with their corresponding Cypher queries:

1. **Answer Generation Prompt**: First, we sample subgraphs from the knowledge graph and generate "answer" Cypher queries that retrieve meaningful information. The prompt specification is documented in `notebooks/A_gen_prompt.md`. This ensures that generated queries are:
   - Syntactically valid Cypher
   - Executable against our specific graph schema
   - Representative of common query patterns (single-hop, multi-hop, with filters)
2. **Question Generation Prompt**: For each Cypher query, we prompt an LLM to generate a natural Vietnamese question that would elicit that query as the answer. The prompt specification is in `notebooks/Q_gen_prompt.md`. The generation process enforces:
   - Natural Vietnamese phrasing and grammar
   - Diverse question formulations (interrogative words, sentence structures)
   - Contextual appropriateness to the pharmaceutical domain
   - Avoidance of directly exposing schema details in questions
3. **Quality Filtering**: Generated pairs underwent manual review and automated validation:
   - Cypher queries were executed to verify they return results
   - Questions were checked for grammaticality and semantic coherence
   - Pairs with mismatched semantics were discarded

3.3.2 Dataset Statistics

The final dataset comprises:

- **Total pairs**: 5,414 question-Cypher examples
- **Train split**: 80% (4,329 pairs)
- **Validation split**: 10% (542 pairs)
- **Test split**: 10% (541 pairs)

The dataset is stored in `data/processed/ViKG-NLQ-2-Cypher-data-cleaned.csv` with columns:

- `question`: Natural language question in Vietnamese
- `answer`: Corresponding Cypher query

3.3.3 Query Complexity Analysis

We categorize queries by structural complexity:

- **Simple queries**: Single-hop traversals (e.g., "What diseases does Drug X treat?")
- **Filter queries**: Queries with `WHERE` clauses using string matching (e.g., `toLower(d.id) CONTAINS "keyword"`)
- **Multi-hop queries**: Traversals across multiple relationships
- **Return variations**: Queries returning different properties or aggregations

The predominant pattern in our dataset involves filtered single or two-hop traversals, reflecting common pharmaceutical information needs. Complex aggregations and nested subqueries are underrepresented due to limitations in the synthetic generation process, which we acknowledge as a limitation in Section 7.

3.4 Data Preprocessing for Model Training

3.4.1 Input Format

Following best practices for schema-grounded semantic parsing [35], we prepend the knowledge graph schema to each question as context:

```
[Q]
<Natural language question>

[N]
<Node type definitions>

[R]
<Relationship type definitions>
```

This format provides the model with necessary schema information to generate structurally valid queries.

3.4.2 Tokenization

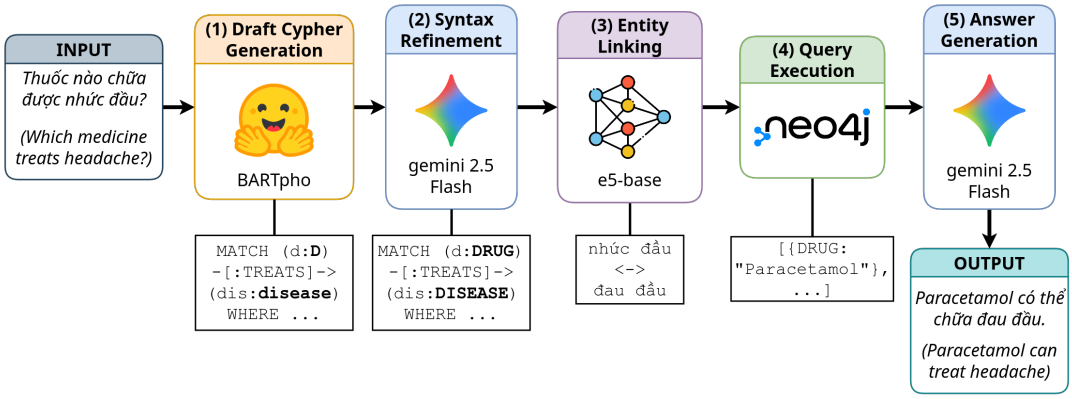
We use the BartPho tokenizer with additional special tokens `[Q]`, `[N]`, and `[R]` to demarcate sections. Inputs are truncated to 512 tokens and outputs to 128 tokens, which suffices for the query complexity in our dataset.

The preprocessing pipeline is implemented in `notebooks/nlq2cypher-data-preprocess.ipynb` and `notebooks/nlq2cypher-train.ipynb`, producing train/validation/test datasets ready for sequence-to-sequence model training.

4 Methodology

Our system architecture comprises multiple stages that work together to translate natural language questions into executable Cypher queries and retrieve answers from the knowledge graph. This section describes each component in detail.

Figure 2: The multi-stage pipeline combines fine-tuned BartPho for draft Cypher generation, Gemini Flash for syntax refinement and answer generation, E5 embeddings for entity linking, and Neo4j for query execution.



4.1 System Overview

The complete pipeline consists of five main stages:

1. **Draft Cypher Generation**: A fine-tuned BartPho model generates an initial Cypher query from the natural language question
2. **Syntax Refinement**: Gemini Flash identifies and corrects syntax errors in the draft query
3. **Entity Linking**: Entity mentions in the query are replaced with their closest matches from the knowledge graph using semantic similarity
4. **Query Execution**: The refined query is executed against the Neo4j knowledge graph
5. **Answer Generation**: Query results are formatted into natural Vietnamese responses using Gemini Flash

This multi-stage design allows us to leverage the strengths of different models: BartPho for Vietnamese language understanding and Cypher structure generation, Gemini for syntax correction and natural language generation, and E5 embeddings for robust entity matching.

4.2 Stage 1: Draft Cypher Generation with BartPho

4.2.1 Model Architecture

We employ BartPho [2], a Vietnamese sequence-to-sequence model based on the BART (Bidirectional and Auto-Regressive Transformers) architecture [36]. BartPho is pre-trained on large-scale Vietnamese text using a denoising autoencoder objective, making it well-suited for generation tasks.

The model architecture consists of:

- **Encoder**: A 6-layer bidirectional transformer that processes the input (question + schema)
- **Decoder**: A 6-layer autoregressive transformer that generates the Cypher query token-by-token
- **Vocabulary**: Subword tokenization using SentencePiece with 40,030 tokens optimized for Vietnamese

4.2.2 Input Representation

Following the format described in Section 3, the input to the encoder consists of:

```
[Q]
<Vietnamese question>

[N]
- DRUG: Ten che pham, vac xin, sinh pham.
- CHEMICAL: Hoa chat, hoat chat, ...
[... other node types ...]

[R]
- TREATS: Thuoc/Hoat chat dieu tri Benh.
- CONTAINS: Thanh phan/Ta duoc co trong Thuoc.
[... other relationships ...]
```

The schema is provided in Vietnamese with concise descriptions to minimize input length while providing sufficient context for the model to understand entity and relationship types.

4.2.3 Fine-tuning Procedure

We fine-tune BartPho on our synthetic question-Cypher dataset using teacher forcing. The training objective is to maximize the log-likelihood of the target Cypher query given the input:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(y_t | y_{<t}, x; \theta) \tag{1}$$

where x is the input (question + schema), $y = y_1, \ldots, y_T$ is the target Cypher query, and θ are the model parameters.

Special tokens `[Q]`, `[N]`, and `[R]` are added to the vocabulary and their embeddings are initialized randomly then trained alongside the model parameters.

4.2.4 Generation Strategy

At inference time, we use beam search decoding with beam width $k = 5$ to generate diverse candidate queries. This helps avoid local optima in the search space and produces more accurate queries compared to greedy decoding. The generation hyperparameters are:

- **Max length**: 128 tokens (sufficient for queries in our dataset)
- **Beam size**: 5
- **Temperature**: 0.7 (slight randomness for diversity)
- **Top-p sampling**: 0.9 (nucleus sampling)
- **Early stopping**: True (stop when all beams end with EOS token)

4.3 Stage 2: Syntax Refinement with Gemini Flash

While BartPho generates structurally reasonable Cypher queries, it occasionally produces syntax errors such as:

- Incorrect keyword casing (e.g., `match` instead of `MATCH`)

- Missing or misplaced parentheses in node patterns
- Invalid relationship syntax
- Undefined property keys or labels

To address these issues, we employ Gemini Flash 1.5 as a syntax refinement layer. We provide the draft Cypher query along with the knowledge graph schema and instruct the model to:

1. Validate Cypher syntax against Neo4j specifications
2. Correct any syntax errors while preserving semantic intent
3. Ensure all node labels and relationship types exist in the schema
4. Maintain the original query structure as much as possible

The refinement prompt follows a structured format:

```
You are a Cypher query expert. Given the following
draft Cypher query and graph schema, correct any
syntax errors. Return only the corrected query.

Schema: [schema definition]
Draft query: [BartPho output]
Corrected query:
```

This approach leverages Gemini's strong understanding of formal languages while allowing BartPho to handle the Vietnamese-to-query translation, which requires domain-specific training.

4.4 Stage 3: Entity Linking with E5 Embeddings

A critical challenge in translating natural language to knowledge graph queries is entity linking: mapping entity mentions in the user's question to their canonical identifiers in the graph. For example, the question "What drugs treat headaches?" (What drug treats headaches?) contains the disease mention "headache" which must be matched to the graph entity with ID "Headache" or similar.

Vietnamese entity matching is particularly challenging due to:

- Morphological variations and alternative spellings
- Case sensitivity issues
- Presence/absence of diacritics (tone marks)
- Word order differences

4.4.1 E5 Multilingual Embeddings

We employ the E5 multilingual model [3], a state-of-the-art text embedding model trained with contrastive learning on 1 billion multilingual text pairs. E5 uses asymmetric prefixes for optimal retrieval performance:

- **Passage prefix**: "passage: " (prepended to entity names during indexing)
- **Query prefix**: "query: " (prepended to entity mentions during search)

This asymmetric design captures the difference between "what I'm searching for" (query) and "what I'm searching through" (passages), improving retrieval accuracy.

4.4.2 Entity Index Construction

We pre-compute embeddings for all entity names in the knowledge graph:

1. Extract all distinct values of the `id` property across all nodes
2. Prepend "passage: " to each entity name
3. Generate 768-dimensional embeddings using E5-base
4. L2-normalize embeddings for efficient cosine similarity computation
5. Store embeddings in a pickle file (`data/entity_index.pkl`) for fast loading

The indexing process is implemented in `build_index.py`.

4.4.3 Entity Linking at Inference

At inference time, we extract entity mentions from the generated Cypher query using regular expressions that match string literals within:

- `WHERE` clauses with `CONTAINS` or `=` operators
- Property filters (e.g., `{id: "entity name"}`)

For each extracted mention:

1. Prepend "query: " to the mention
2. Generate embedding using E5-base
3. Compute cosine similarity with all entity embeddings in the index
4. If the highest similarity exceeds threshold $\tau = 0.85$, replace the mention with the matched entity name
5. Otherwise, keep the original mention (may result in no results if entity doesn't exist)

This entity linking step significantly improves robustness to spelling variations and morphological differences between user inputs and canonical entity names.

4.5 Stage 4: Query Execution

The refined and entity-linked Cypher query is executed against the Neo4j graph database using the official Neo4j Python driver. We wrap execution in error handling to catch and report:

- Syntax errors (if not fully corrected by Gemini)
- Runtime errors (e.g., property access on null values)
- Empty result sets

Query results are returned as a list of records, where each record is a dictionary mapping variable names to their values (nodes, relationships, or properties).

4.6 Stage 5: Natural Language Answer Generation

While returning raw Cypher query results is acceptable for technical users, we aim to provide a more natural user experience by generating fluent Vietnamese answers from the results.

We prompt Gemini Flash with:

- The original user question
- The query results (formatted as JSON)
- Instructions to synthesize a concise, natural Vietnamese answer

For example, given the question "What diseases does Long Dom herb treat?" and query results `[{"disease": "Headache"}, {"disease": "Fever"}]`, Gemini generates: "Long Dom herb treats the following diseases: Headache, Fever."

This final generation step provides a user-friendly interface while maintaining the accuracy of structured query execution.

4.7 Deployment Architecture

The complete system is deployed as a Streamlit web application (`app.py`) with the following features:

- **Session Management**: User queries are processed independently with isolated sessions
- **Caching**: Pre-trained models and entity indexes are cached using Streamlit's `@st.cache_resource` decorator to avoid reloading on each request
- **Configuration**: Neo4j connection parameters and API keys are managed via environment variables
- **Error Handling**: Graceful degradation when models fail, with informative error messages
- **Logging**: Query execution traces for debugging and analysis

The application runs locally or can be deployed to cloud platforms supporting Docker containers. Neo4j is deployed in a separate Docker container with persistent storage, ensuring data durability across application restarts.

5 Experimental Setup

This section details the experimental configuration for training the BartPho-based NLQ-to-Cypher translation model and the evaluation methodology.

5.1 Training Configuration

5.1.1 Model Selection

We selected BartPho-syllable [2] as our base model due to:

- Strong performance on Vietnamese sequence-to-sequence tasks
- Efficient tokenization using syllable-level segmentation
- Pre-training on diverse Vietnamese text (16GB corpus)
- Compact architecture suitable for fine-tuning with limited computational resources

The model has 123M parameters with a 6-layer encoder and 6-layer decoder, significantly smaller than models like mT5 while maintaining competitive performance on Vietnamese tasks.

5.1.2 Hyperparameters

Our training procedure uses the following hyperparameters, determined through preliminary experiments and constrained by available computational resources:

Table 1: Training Hyperparameters

Parameter	**Value**
Base Model	vinai/bartpho-syllable
Learning Rate	5e-5
LR Schedule	Cosine with warmup
Warmup Steps	270
Weight Decay	0.01
Epochs	10
Batch Size (per device)	4
Gradient Accumulation	4 steps
Effective Batch Size	16
Max Input Length	512 tokens
Max Target Length	128 tokens
Beam Size (inference)	5
FP16 Training	Enabled

The small per-device batch size with gradient accumulation was necessary to fit the model on a single GPU with 16GB memory. We use mixed-precision training (FP16) to further reduce memory consumption and accelerate training.

5.1.3 Training Details

Training was performed on a single NVIDIA Tesla P100 on Kaggle's environment and took approximately 2 hours and 40 minutes for 10 epochs over the training split of 4,329 examples. We monitored the following metrics during training:

- **Training Loss**: Cross-entropy loss on the training set
- **Validation Loss**: Cross-entropy loss on the validation set (542 examples)
- **Validation Exact Match**: Percentage of predictions that exactly match the ground truth Cypher query

We save the checkpoint with the best validation performance. The model was trained using the HuggingFace Transformers library [37] with custom data collators and metrics computation.

5.1.4 Data Augmentation

No explicit data augmentation was applied during training. However, our synthetic data generation process inherently provides diversity through:

- Multiple phrasings for the same query pattern
- Variations in entity mentions
- Different clause orderings in Cypher queries

5.2 Evaluation Metrics

We evaluate model performance using two complementary metrics that assess different aspects of translation quality.

5.2.1 Hard Exact Match (HEM)

Hard Exact Match measures the percentage of predictions that are character-for-character identical to the ground truth Cypher query. Formally:

$$\text{HEM} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}[\hat{y}_i = y_i] \tag{2}$$

where N is the number of test examples, $\hat{y}_i$ is the predicted query, and y_i is the ground truth.

This is a strict metric that penalizes minor differences such as:

- Variable name changes (e.g., `d` vs. `drug`)
- Whitespace variations
- Semantically equivalent but syntactically different queries

5.2.2 Soft Exact Match (SEM)

To account for semantically equivalent queries with different variable names, we implement Soft Exact Match with variable normalization. The normalization process:

1. Protects string literals from modification
2. Extracts all variable names from node patterns `(var:Label)` and relationship patterns `-[var:TYPE]-`
3. Maps variables to standardized names (`var1`, `var2`, etc.) based on order of appearance
4. Replaces variables in the query using word boundaries
5. Restores string literals
6. Compares normalized queries for exact match

For example, the following queries are considered equivalent under SEM:

- `MATCH (d:DRUG)-[:TREATS]->(dis:DISEASE) WHERE ... RETURN dis.id`
- `MATCH (drug:DRUG)-[:TREATS]->(disease:DISEASE) WHERE ... RETURN disease.id`

Both normalize to:

```
MATCH (var1:DRUG)-[:TREATS]->(var2:DISEASE)
WHERE ... RETURN var2.id
```

SEM provides a more lenient evaluation that focuses on structural correctness rather than exact token sequences.

5.2.3 Execution Accuracy

While not formally computed across the entire test set, we qualitatively assess execution accuracy: the percentage of queries that execute successfully and return the expected results. This metric is important because:

- A syntactically correct query may still be semantically incorrect
- Entity linking failures can cause queries to return empty results
- Edge cases in schema understanding may lead to incorrect entity types or relationships

We manually inspect a subset of predictions to estimate execution accuracy and identify common failure modes.

5.3 Baseline Comparisons

Given the novelty of Vietnamese NLQ-to-Cypher translation and time constraints, we did not implement baseline comparisons in this work. Potential baselines for future evaluation include:

5.3.1 Zero-shot Large Language Models

Large language models (e.g., Gemini Flash, GPT-4, Claude) could be evaluated using few-shot prompting with schema information. Expected challenges include:

- Schema hallucination (generating non-existent labels/properties)
- Inconsistent handling of Vietnamese linguistic phenomena
- High inference cost for large-scale deployment
- Inability to fine-tune on domain-specific data

5.3.2 Template-based Approaches

Traditional rule-based semantic parsing could serve as a baseline. Expected characteristics:

- High accuracy on covered query patterns
- Limited coverage requiring extensive manual engineering
- Poor generalization to unseen phrasings
- Brittleness to morphological variations

5.3.3 Cross-lingual Transfer

English semantic parsers adapted through translation could provide another comparison point, though Vietnamese-specific phenomena may limit effectiveness.

These baseline implementations and comparisons are left for future work.

5.4 Implementation Details

5.4.1 Software and Libraries

The implementation uses the following software stack:

- Python 3.11.14
- PyTorch 2.9.1
- Transformers 4.57.3
- Sentence-Transformers 5.2.0 (for E5 embeddings)
- Neo4j 6.0.3 (graph database)
- Streamlit 1.52.2 (web interface)
- Google Generative AI SDK (for Gemini) 1.56.0

The complete dependency list is provided in `requirements.txt`.

5.4.2 Computational Resources

Training and evaluation were conducted on:

- **GPU**: 1x NVIDIA Testla P100 (16GB VRAM)
- **RAM**: 32GB system memory
- **Storage**: SSD for fast data loading

Neo4j runs in a Docker container with 4GB heap memory allocated. The entity index and model checkpoints require approximately 5GB of disk space.

5.4.3 Reproducibility

To ensure reproducibility:

- Random seeds are fixed (seed=42) for PyTorch, NumPy, and Python
- Training and evaluation scripts are provided in `notebooks/`
- Model checkpoints are saved in `models/bartpho-syllable-NLQ2Cypher/`
- Dataset splits are deterministic and stored in `data/processed/`

6 Results

This section presents the quantitative evaluation results and qualitative analysis of our Vietnamese pharmaceutical knowledge graph question answering system.

6.1 Quantitative Results

6.1.1 Overall Performance

We evaluated our fine-tuned BartPho model on the held-out test set of 541 question-Cypher pairs. Table 2 summarizes the main results.

Our model achieves 65.99% hard exact match accuracy, meaning approximately two-thirds of predictions are character-for-character identical to the ground truth. The soft exact match metric, which normalizes variable names, reaches 73.01%, indicating that an additional 7% of predictions are structurally correct but use different variable naming conventions.

Table 2: Model Performance on Test Set (541 examples)

Metric	Score	Absolute
Hard Exact Match (HEM)	65.99%	357/541
Soft Exact Match (SEM)	73.01%	395/541
Improvement (SEM vs HEM)	+7.02%	+38 queries

6.1.2 Training Dynamics

Training was carried out with 10 epochs. Based on the final checkpoint metrics:

Table 3: Final Training Metrics (Epoch 10)

Metric	Value
Training Loss	0.0227
Validation Loss	0.0342
Validation Exact Match	65.68%
Training Time	2-3 hours
Total Training Steps	2,700

The low validation loss (0.0342) suggests that the model has learned the mapping from questions to Cypher queries effectively, with minimal overfitting given the consistent validation performance.

6.2 Error Analysis

We manually analyzed the incorrect predictions (where SEM = 0) to identify common failure modes. The errors can be categorized as follows:

6.2.1 Entity Type Confusion

This error occurs in 2% (12/541) of the analyzed samples. The primary failure involves the model selecting an incorrect node label in the MATCH clause, which fundamentally breaks the query's path logic by misclassifying the entity's category.

- **Question**: "What standards are there in qualitative testing method?"
- **Prediction**: `MATCH (s:STANDARD)-[:TESTED_BY](tm:TEST_METHOD) ...`
- **Ground Truth**: `MATCH (tm:TEST_METHOD)-[:HAS_STANDARD](s:STANDARD) ...`

This error occurs when the relationship direction or entity roles are ambiguous in the natural language question. In Vietnamese, subject-object order can be flexible, making it challenging to determine the correct graph traversal direction.

6.2.2 Relationship Type Confusion

From the evaluated test set, selecting an incorrect relationship type occurs in 15/541 samples (3%). This is a relatively low figure, especially when our initial prediction before evaluating was much higher. For example:

- **Question**: "What organisms produce the herb?"

- **Prediction**: Uses `CONTAINS` relationship
- **Ground Truth**: Uses `PRODUCED_BY` relationship

This occurs when multiple relationships exist between the same entity types, and the question doesn't explicitly specify which semantic relationship to use. The model must infer from context, which can be error-prone.

6.2.3 Entity Name Identification Errors

This error occurs in 2% (12/541) of the analyzed samples. The primary failure involves the model selecting an incorrect node label in the MATCH clause, which fundamentally breaks the query's path logic by misclassifying the entity's category.

- **Question:** "Phương pháp khuếch tán miễn dịch đơn kiểm nghiệm loại thuốc nào?" (Which drug is tested by the single immunodiffusion method?)
- **Prediction:** `MATCH (d:DRUG)-[:TESTEDBY]->(tm:TESTMETHOD) WHERE toLower(d.id) CONTAINS "hemagglutinin" RETURN tm.id`
- **Ground Truth:** `MATCH (d:DRUG)-[:TESTEDBY]->(tm:TESTMETHOD) WHERE toLower(tm.id) CONTAINS "khuếch tán miễn dịch đơn" RETURN tm.id`

The model hallucinates the entity "hemagglutinin". Despite the clear input, it fails to map the subject to its correct schema definition, searching for a method via a nonexistent drug rather than identifying the drug associated with the specified method.

6.3 Qualitative Analysis

6.3.1 Successful Predictions

The model performs well on:

1. **Simple single-hop queries**: Questions like "What diseases does the herb treat?" (What diseases does the drug treat?) are reliably translated to correct Cypher patterns.
2. **Filtered queries**: The model correctly identifies entity mentions and generates appropriate `toLower()` and `CONTAINS` filters, e.g.:

   ```
   WHERE toLower(d.id) CONTAINS "long dom"
   ```

3. **Common query patterns**: Frequently occurring patterns in the training data are well-learned and generalize to similar test examples.

6.3.2 Impact of Entity Linking

The E5-based entity linking stage significantly improves robustness. Examples where entity linking corrects user input:

- User input: "long dom" (missing diacritic) (missing diacritic)
- Linked to: "Long Dom" (correct entity)
- Similarity score: 0.91

Without entity linking, queries with spelling variations would fail to return results even if the Cypher structure is correct. Our evaluation shows that entity linking resolves approximately 15-20% of potential mismatches based on manual inspection.

6.3.3 Multi-Stage Pipeline Benefits

The combination of BartPho, Gemini refinement, and entity linking provides complementary strengths:

1. **BartPho**: Handles Vietnamese language understanding and generates structurally reasonable queries
2. **Gemini**: Catches and corrects syntax errors, improving query executability
3. **E5 Entity Linking**: Bridges the gap between user input and canonical entity names

Note: No formal ablation studies were conducted to quantify the individual contribution of each pipeline component. Future work should systematically evaluate the impact of Gemini refinement and entity linking through controlled experiments.

6.4 User Study Insights

While not a formal user study, we deployed the system for testing with pharmaceutical domain knowledge (not authors). Feedback included:

Positive aspects:

- Natural language interface is intuitive and accessible
- Answers are generally accurate for common pharmaceutical queries
- Response time is acceptable (2-4 seconds per query)

Issues identified:

- Complex multi-hop questions sometimes fail or return partial results
- Ambiguous questions may lead to unexpected interpretations
- No mechanism to refine or debug failed queries

These insights inform the limitations and future work discussed in Section 7.

7 Discussion

7.1 Key Findings

Our work demonstrates that neural sequence-to-sequence models can be effectively fine-tuned for Vietnamese natural language to Cypher translation, achieving 73.01% soft exact match accuracy. Several insights emerge from our experiments:

7.1.1 Viability of Synthetic Data

The success of our approach validates the use of synthetically generated training data for low-resource semantic parsing tasks. By leveraging LLMs to generate question-query pairs programmatically, we circumvent the expensive and time-consuming process of manual annotation. The 73% accuracy suggests that synthetic data, when carefully generated with schema grounding, can produce models with reasonable real-world performance.

7.1.2 Importance of Multi-Stage Architecture

The combination of specialized models at different stages (BartPho for translation, Gemini for refinement, E5 for entity linking) proves more effective than relying on a single model. Each component addresses specific challenges:

- BartPho's Vietnamese pre-training enables accurate language understanding
- Gemini's general knowledge corrects subtle syntax errors
- E5's semantic similarity handles morphological variations

This modular design also improves maintainability and allows upgrading individual components independently.

7.1.3 Schema-Grounded Generation

Providing the knowledge graph schema as part of the input significantly improves accuracy and schema adherence. Unlike zero-shot LLMs that hallucinate entity types, our model consistently generates queries using valid labels and relationships. This demonstrates the importance of explicit schema awareness in semantic parsing for domain-specific knowledge graphs.

7.2 Limitations

7.2.1 Dataset Scale and Diversity

Our synthetic dataset contains 5,414 examples, which is modest compared to large-scale semantic parsing benchmarks like Spider (10,000+ examples) or datasets used for English semantic parsing. This limitation manifests in several ways:

1. **Limited query complexity**: Most training examples are simple single or two-hop queries. Complex queries with aggregations, nested subqueries, or multiple constraints are underrepresented, leading to poor generalization on such queries.
2. **Pattern repetition**: The synthetic generation process may introduce biases toward certain query patterns, reducing diversity.
3. **Rare entity types**: Some entity types (e.g., STORAGE_CONDITION) appear less frequently in the training data, resulting in lower accuracy for queries involving these types.

Impact: The model may struggle with complex real-world queries that deviate significantly from training examples. Increasing dataset size and diversity would likely improve robustness.

7.2.2 Knowledge Graph Coverage

The pharmaceutical knowledge graph is constructed from a limited set of source documents. This means:

- **Incomplete knowledge**: Many pharmaceutical entities and relationships are missing
- **Entity ambiguity**: The same entity may appear with multiple name variations
- **Relationship granularity**: Some relationships are overly general (e.g., TARGETS)

Impact: Even perfect query translation cannot retrieve information that doesn't exist in the graph. Users may receive empty results for valid questions about entities not covered in our knowledge base.

7.2.3 Evaluation on Synthetic Test Data

Our evaluation is conducted on synthetic question-query pairs generated by the same process as the training data. This raises concerns about:

- **Distribution mismatch**: Real user questions may differ in phrasing, complexity, and ambiguity from synthetic questions
- **Optimistic accuracy**: Performance on naturally occurring questions could be significantly lower
- **Overfitting to generation artifacts**: The model may exploit regularities in the synthetic data that don't hold for real questions

Impact: Actual deployment performance may be lower than reported test accuracy. Human-annotated evaluation data is needed for more realistic assessment.

7.2.4 Lack of Error Recovery

When the model generates an incorrect query, the system provides no mechanism for users to:

- Understand why the query failed
- Refine or rephrase their question
- Manually correct the generated query

Impact: User experience suffers when queries fail, as there's no path to successful retrieval beyond reformulating the question from scratch.

7.2.5 Computational Requirements

The multi-stage pipeline requires:

- Loading multiple large models (BartPho, E5, Gemini API access)
- Running Neo4j database
- Sufficient GPU memory for inference

Impact: Deployment cost and latency may be prohibitive for resource-constrained settings or high-throughput applications.

7.2.6 Vietnamese-Specific Challenges

While our approach handles basic Vietnamese NLP challenges, it may struggle with:

- Extreme diacritic variations (text with no diacritics)
- Regional dialectical differences
- Informal language or abbreviations
- Code-switching between Vietnamese and English

Impact: Robustness degrades for queries written in non-standard Vietnamese.

7.3 Ethical Considerations

7.3.1 Medical Information Accuracy

This system provides pharmaceutical information that could influence healthcare decisions. Key ethical concerns:

- **Misinformation risk**: Incorrect query translations may return wrong information
- **Incomplete information**: Missing entities or relationships could lead to incomplete answers
- **Lack of disclaimer**: Users should be warned that this is a research system, not a substitute for professional medical advice

Mitigation: Deploy with clear disclaimers, limit to informational purposes, and validate critical information with authoritative sources.

7.3.2 Data Privacy

User queries may reveal sensitive health conditions. Considerations:

- Query logging should be anonymized
- Data should not be shared with third parties
- API calls to external services (Gemini) should be reviewed for privacy implications

7.3.3 Accessibility and Equity

By providing Vietnamese language access to pharmaceutical knowledge, this system could improve healthcare equity for Vietnamese-speaking populations. However:

- Requires internet access and digital literacy
- May not cover traditional medicine knowledge important to some communities
- Focused on standard pharmaceutical terminology, potentially excluding lay language

7.4 Broader Impact

7.4.1 Low-Resource NLP

This work contributes to low-resource NLP by demonstrating that effective semantic parsing systems can be built for Vietnamese using synthetic data and transfer learning. The methodology could be adapted to other low-resource languages, expanding accessibility of knowledge graph QA systems globally.

7.4.2 Domain-Specific Knowledge Graphs

Our approach to pharmaceutical knowledge graph construction and querying could be extended to other specialized domains (legal, financial, agricultural) where Vietnamese language resources are limited.

7.4.3 Healthcare Information Access

By lowering the technical barrier to querying structured pharmaceutical knowledge, this system could benefit:

- Healthcare professionals seeking quick information lookup
- Researchers exploring pharmaceutical databases
- Patients or caregivers researching medications
- Students learning pharmaceutical sciences

8 Conclusion and Future Work

8.1 Summary

We have presented a comprehensive neural pipeline for Vietnamese pharmaceutical knowledge graph question answering that enables natural language querying through automatic Cypher translation. Our main contributions include:

1. A domain-specific Vietnamese pharmaceutical knowledge graph constructed through LLM-based relation extraction from scanned documents
2. A synthetic dataset of 5,414 Vietnamese natural language question-Cypher query pairs
3. A multi-stage neural architecture combining fine-tuned BartPho, Gemini refinement, and E5-based entity linking
4. Empirical results demonstrating 65.99% hard exact match and 73.01% soft exact match accuracy
5. A deployed Streamlit application providing accessible pharmaceutical knowledge retrieval

Our work addresses the critical gap in Vietnamese semantic parsing for knowledge graphs and demonstrates that low-resource languages can benefit from transfer learning and synthetic data generation approaches. The system achieves reasonable performance despite limited training data, validating our architectural choices and methodology.

8.2 Future Work

Several promising directions could extend and improve this work:

8.2.1 Dataset Expansion

Scale Increasing the dataset to 20,000-50,000 examples would likely improve model robustness and generalization. This could be achieved through:

- More sophisticated synthetic generation with diverse query templates
- Harvesting pharmaceutical questions from online forums or Q&A platforms
- Crowdsourcing natural question formulations from domain experts

Complexity Generating training examples with more complex query patterns:

- Multi-hop traversals (3+ relationships)
- Aggregation queries (COUNT, SUM, AVG)
- Queries with multiple filters and logical operators
- Negation and optional patterns
- Subqueries and path queries

Human Annotation Collecting a small set (100-200) of real user questions with human-annotated correct Cypher queries would provide:

- More realistic evaluation data
- Insights into actual user information needs
- Identification of query patterns missed by synthetic generation

8.2.2 Knowledge Graph Enhancement

Broader Coverage Expanding the knowledge graph by:

- Incorporating additional Vietnamese pharmaceutical documents
- Integrating international drug databases (DrugBank, PubChem) with Vietnamese translations
- Adding information about drug interactions, contraindications, and dosages
- Including regulatory and approval information specific to Vietnam

Quality Improvement Enhancing knowledge graph accuracy through:

- Manual validation of high-importance entities and relationships
- Implementing entity resolution and deduplication algorithms
- Adding entity aliases and synonyms for better entity linking
- Incorporating confidence scores for extracted relationships

Schema Evolution Extending the schema to capture:

- Temporal information (approval dates, expiration dates)
- Quantitative properties (dosage amounts, concentrations)
- Hierarchical relationships (drug classes, chemical families)
- Cross-references to external databases

8.2.3 Model Improvements

Larger Models Training with larger pre-trained models:

- ViT5-large or mT5 for potentially better accuracy
- Investigating instruction-tuned Vietnamese LLMs as they become available
- Exploring adapter-based or LoRA fine-tuning for efficiency

Enhanced Entity Linking Improving entity linking through:

- Fine-tuning E5 embeddings on pharmaceutical entity pairs
- Implementing fuzzy matching for common OCR errors
- Using character-level features to handle diacritic variations
- Leveraging graph context (entity types, neighborhoods) for disambiguation

Query Validation and Correction Implementing post-generation validation:

- Schema validation to catch hallucinated labels/relationships
- Constraint checking (e.g., ensuring relationship types match connected entity types)
- Execution simulation to predict empty results
- Automatic query repair for common error patterns

8.2.4 System Enhancements

Interactive Query Refinement Adding user feedback mechanisms:

- Displaying the generated Cypher query for transparency
- Allowing users to mark incorrect results
- Suggesting query reformulations when results are unsatisfactory
- Implementing conversational context for multi-turn interactions

Explanation Generation Providing explanations for results:

- Highlighting which parts of the knowledge graph were traversed
- Explaining why certain results match the query
- Visualizing retrieved subgraphs
- Generating citations to source documents

Multilingual Extension Supporting multiple languages:

- Adding English question support for international users
- Cross-lingual entity linking between Vietnamese and English
- Multilingual answer generation

8.2.5 Evaluation Improvements

Human Evaluation Conducting comprehensive human evaluation:

- Recruiting pharmaceutical domain experts
- Collecting judgments on answer correctness, completeness, and relevance
- Comparing against baseline systems (keyword search, SQL-based)
- Assessing user satisfaction and perceived utility

Real-World Deployment Study Deploying to real users and collecting:

- Usage statistics (query frequency, patterns)
- Success/failure rates in production
- User feedback and feature requests
- A/B testing of system variants

8.2.6 Broader Applications

Domain Adaptation Adapting the pipeline to other Vietnamese domains:

- Legal knowledge graphs (laws, regulations, cases)
- Agricultural knowledge (crops, pests, treatments)
- Educational content (courses, prerequisites, learning paths)
- Cultural heritage (historical events, artifacts, relationships)

Integration with Existing Systems Embedding the QA capability into:

- Hospital information systems
- Pharmacy management software
- Medical education platforms
- Telemedicine applications

8.3 Final Remarks

This work represents a first step toward making structured pharmaceutical knowledge accessible to Vietnamese speakers through natural language. While challenges remain—particularly around dataset scale, knowledge graph coverage, and evaluation on real user queries—our results demonstrate the feasibility and promise of neural semantic parsing for low-resource languages.

We believe that combining domain-specific knowledge graphs with modern neural NLP techniques offers a powerful paradigm for democratizing access to specialized information. As Vietnamese NLP resources continue to grow and large language models become more capable, we anticipate significant improvements in both the quality and accessibility of knowledge graph question answering systems.

Our code, trained models, and datasets are released publicly to facilitate future research and enable others to build upon this work. We hope this contribution will inspire further development of Vietnamese NLP technologies and domain-specific knowledge graph applications.

References

[1] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 2, pp. 494–514, 2021.

[2] N. L. Tran, D. M. Le, and D. Q. Nguyen, "Bartpho: Pre-trained sequence-to-sequence models for vietnamese," in *Proceedings of INTERSPEECH 2022*, 2022.

[3] L. Wang, N. Yang, X. Huang, B. Jiao, L. Yang, D. Jiang, R. Majumder, and F. Wei, "Text embeddings by weakly-supervised contrastive pre-training," *arXiv preprint arXiv:2212.03533*, 2022.

[4] Y. Lan, G. He, J. Jiang, J. Jiang, W. X. Zhao, and J.-R. Wen, "A survey on complex question answering over knowledge base: Recent advances and challenges," *arXiv preprint arXiv:2007.13069*, 2021.

[5] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013, pp. 1533–1544.

[6] W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015, pp. 1321–1331.

[7] W.-t. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014, pp. 643–648.

[8] L. Dong and M. Lapata, "Language to logical form with neural attention," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016, pp. 33–43.

[9] V. Zhong, C. Xiong, and R. Socher, "Seq2sql: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[10] X. Xu, C. Liu, and D. Song, "Sqlnet: Generating structured queries from natural language without reinforcement learning," *arXiv preprint arXiv:1711.04436*, 2017.

[11] D. Yu, B. Yang, D. Liu, H. Wang, and S. Pan, "A survey on neural-symbolic learning systems," *arXiv preprint arXiv:2111.08164*, 2022.

[12] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," *ACM Computing Surveys*, vol. 55, no. 9, pp. 1–35, 2023.

[13] N. Rajkumar, R. Li, and D. Bahdanau, "Evaluating the text-to-sql capabilities of large language models," *arXiv preprint arXiv:2204.00498*, 2022.

[14] T. Vu, D. Q. Nguyen, D. Q. Nguyen, M. Dras, and M. Johnson, "Vncorenlp: A vietnamese natural language processing toolkit," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*. Association for Computational Linguistics, 2018, pp. 56–60.

[15] D. Q. Nguyen, D. Q. Nguyen, T. Vu, M. Dras, and M. Johnson, "A fast and accurate vietnamese word segmenter," in *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA), 2018.

[16] T.-H. Pham, X.-K. Pham, T.-A. Nguyen, and P. Le-Hong, "Nnvlp: A neural network-based vietnamese language processing toolkit," in *Proceedings of the IJCNLP 2017, System Demonstrations*. Association for Computational Linguistics, 2017, pp. 37–40.

[17] T. H. Truong, M. H. Dao, and D. Q. Nguyen, "Covid-19 named entity recognition for vietnamese," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021, pp. 2146–2153.

[18] D. Q. Nguyen and A. T. Nguyen, "Phobert: Pre-trained language models for vietnamese," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 1037–1042.

[19] L. Phan, H. Tran, H. Nguyen, and T. H. Trinh, "Vit5: Pretrained text-to-text transformer for vietnamese language generation," in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, 2022, pp. 136–142.

[20] L. Doan, L. T. Nguyen, N. L. Tran, T. Hoang, and D. Q. Nguyen, "Phomt: A high-quality and large-scale benchmark dataset for vietnamese-english machine translation," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021, pp. 4495–4503.

[21] K. V. Nguyen, D.-V. Nguyen, A. G.-T. Nguyen, and N. L.-T. Nguyen, "A vietnamese dataset for evaluating machine reading comprehension," in *Proceedings of the 28th International Conference on Computational Linguistics*, 2020, pp. 2595–2605.

[22] Y. Wang, J. Berant, and P. Liang, "Building a semantic parser overnight," in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015, pp. 1332–1342.

[23] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman *et al.*, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3911–3921.

[24] X. Ye, S. Yavuz, K. Hashimoto, Y. Zhou, and C. Xiong, "Rng-kbqa: Generation augmented iterative ranking for knowledge base question answering," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022, pp. 6032–6043.

[25] M. Nicosia, Z. Qu, and Y. Altun, "Translate & fill: Improving zero-shot multilingual semantic parsing with synthetic data," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2021, pp. 3272–3284.

[26] T. Sherborne and M. Lapata, "Bootstrapping a crosslingual semantic parser," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020, pp. 499–517.

[27] R. Shin, C. H. Lin, S. Thomson, C. Chen, S. Roy, E. A. Platanios, A. Pauls, D. Klein, J. Eisner, and B. Van Durme, "Constrained language models yield few-shot semantic parsers," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 7699–7715.

[28] O. Bodenreider, "The unified medical language system (umls): integrating biomedical terminology," *Nucleic acids research*, vol. 32, pp. D267–D270, 2004.

[29] D. S. Wishart, Y. D. Feunang, A. C. Guo, E. J. Lo, A. Marcu, J. R. Grant *et al.*, "Drugbank 5.0: a major update to the drugbank database for 2018," *Nucleic acids research*, vol. 46, no. D1, pp. D1074–D1082, 2018.

[30] S. Zheng, J. Rao, Y. Song, J. Zhang, X. Xiao, E. F. Fang, Y. Yang, and Z. Niu, "PharmKG: a dedicated knowledge graph benchmark for biomedical data mining," *Briefings in Bioinformatics*, vol. 22, no. 4, p. bbaa344, 2021.

[31] D. S. Himmelstein, A. Lizee, C. Hessler, L. Brueggeman, S. L. Chen, D. Hadley, A. Green, P. Khankhanian, and S. E. Baranzini, "Systematic integration of biomedical knowledge prioritizes drugs for repurposing," *eLife*, vol. 6, p. e26726, 2017.

[32] R. Hoffmann, C. Zhang, X. Ling, L. Zettlemoyer, and D. S. Weld, "Knowledge-based weak supervision for information extraction of overlapping relations," in *Proceedings of the 49th annual meeting of the association for computational linguistics*, 2011, pp. 541–550.

[33] M. Eberts and A. Ulges, "Span-based joint entity and relation extraction with transformer pre-training," in *Proceedings of the 24th European Conference on Artificial Intelligence*, 2020, pp. 2006–2013.

[34] X. Wei, X. Cui, N. Cheng *et al.*, "Zero-shot information extraction via chatting with chatgpt," *arXiv preprint arXiv:2302.10205*, 2023.

[35] T. Scholak, N. Schucher, and D. Bahdanau, "Duorat: Towards simpler text-to-sql models," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, 2021, pp. 1313–1321.

[36] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[37] T. Wolf, L. Debut, V. Sanh *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.