Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)
Кафедра _____Системотехніки_____
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____Дослідження і розробка системи пошуку даних клінічних випробувань з використанням відкритих API_____

_____Research on the Development of a Search System for Clinical Trial Data Using Open APIs_____

(тема)


Виконала:
Студентка _2_ курсу, групи _____ІТПм-24-1_____
_____Марія Чергинська_____
(власне ім'я, прізвище)
Спеціальність ____122 Комп'ютерні науки____
(код і повна назва спеціальності)
Тип програми _____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _інформаційні технології проектування_____
(повна назва освітньої програми)

Керівник _____доцент Ситнікова П.Е.___
(посада, прізвище, ініціали)


Допускається до захисту
Зав. кафедри системотехніки _____ ___Гребенік І.В.___
(підпис) (прізвище, ініціали

2025 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерних наук_____

Кафедра_____системотихніки_____

Рівень вищої освіти_____перший (бакалаврський)_____

Спеціальність_____122 Комп'ютерні науки_____
(код і повна назва)

Тип програми _____освітньо-професійна_____

Освітня програма_____комп'ютерні науки та технології_____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____ 20 25_ р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентці____Чергинській Марії Дмитрівні_____

1. Тема роботи ___Research on the Development of a Search System for Clinical Trial Data Using Open APIs
затверджена наказом університету від «24» листопада 2025р. № __1058_
2. Термін подання студентом роботи до екзаменаційної комісії 18 грудня 2025 р.
3. Вихідні дані до роботи __природномовні клінічні запити користувачів; дані клінічних досліджень з бази ClinicalTrials.gov; структуровані медичні терміни та семантичні зв'язки (MeSH, UMLS, CliniFact), структуровані JSON-запити, сумісні з ClinicalTrials.gov API; результати пошуку клінічних досліджень; аналітичні візуалізації та експортовані дані у форматах PDF, CSV, JSON.
4. Перелік питань, що потрібно опрацювати в роботі___Вступ, Аналіз предметної області та постановка задачі дослідження, Математичні основи та архітектура системи, Розробка системи пошуку клінічних досліджень, Експериментальні дослідження та аналіз результатів.
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) схема високорівневої архітектури системи пошуку клінічних досліджень, схеми взаємодії між модулями системи (NLP-модуль, API-інтеграція, вебінтерфейс), діаграми процесу трансформації природномовного запиту у структурований формат, графіки розподілу клінічних досліджень за фазами, статусами та географією, таблиці з результатами

експериментального порівняння моделей LLM та методів PEFT, скріншот вебінтерфейсу системи

# КАЛЕНДАРНИЙ ПЛАН

| | Назва етапів роботи | Терміни виконання етапів | Примітка |
|---|---|---|---|
| | Аналіз предметної області та існуючих систем пошуку клінічних досліджень | 01.10.2025 – 10.10.2025 | Виконано |
| | Аналіз технологій обробки природної мови та великих мовних моделей у медичній сфері | 10.10.2025 – 18.10.2025 | Виконано |
| | Формування та анотування навчального набору даних для задачі трансформації клінічних запитів | 18.10.2025 – 28.10.2025 | Виконано |
| | Проєктування архітектури системи пошуку клінічних досліджень | 28.10.2025 – 02.11.2025 | Виконано |
| | Імплементація модуля обробки природномовних запитів з використанням LLM та PEFT | 02.11.2025 – 15.11.2025 | Виконано |
| | Реалізація інтеграції з ClinicalTrials.gov API та обробки структурованих відповідей | 15.11.2025 – 22.11.2025 | Виконано |
| | Розробка вебінтерфейсу та інтерактивних візуалізацій результатів пошуку | 22.11.2025 – 02.12.2025 | Виконано |
| | Проведення експериментальних досліджень та порівняльний аналіз моделей і PEFT-методів | 02.12.2025 – 10.12.2025 | Виконано |
| | Аналіз результатів, формування висновків та оформлення кваліфікаційної роботи | 10.12.2025 – 15.12.2025 | Виконано |
| | Захист кваліфікаційної роботи в екзаменаційній комісії | 24.12.2025 | Виконано |

Дата видачі завдання 24 листопада 2025р.

Студент _____
(підпис)

Керівник роботи _____ доцент Ситнікова П.Е.
(підпис) (посада, власне ім'я, прізвище)

.

*Я, як студентка ХНУРЕ, розумію і підтримую політику закладу із академічної доброчесності. Я не надававала і не одержувала недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.*

24.12.2025

*Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування.*

Керівник кваліфікаційної роботи _____ доцент Ситнікова П.Е.

*Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні.*

Керівник кваліфікаційної роботи _____ доцент Ситнікова П.Е.

*Попередній захист проведено 24 грудня 2025 р.*

Керівник кваліфікаційної роботи _____ доцент Ситнікова П.Е.

# РЕФЕРАТ

Кваліфікаційна робота: 86с., 8 рис., 6 табл., 25 джерел, 2 додатки.

КЛІНІЧНІ ВИПРОБУВАННЯ, ОБРОБКА ПРИРОДНОЇ МОВИ, ВЕЛИКІ МОВНІ МОДЕЛІ, PEFT, CLINICALTRIALS.GOV, ВЕБЗАСТОСУНОК, МЕДИЧНА ІНФОРМАТИКА.

Об’єктом дослідження є процес пошуку та аналізу даних клінічних випробувань у відкритих медичних інформаційних системах.

Предметом дослідження є методи та програмні засоби трансформації природномовних запитів у структуровані параметри пошуку клінічних досліджень із використанням великих мовних моделей.

Метою роботи є розробка та дослідження інтелектуальної системи пошуку клінічних випробувань на основі великих мовних моделей із параметрично ефективним донавчанням, а також оцінювання ефективності різних архітектур і методів PEFT для задачі перетворення природномовних медичних запитів.

Методами дослідження є: системний аналіз, методи обробки природної мови, машинне навчання, параметрично ефективне донавчання великих мовних моделей, експериментальне порівняльне оцінювання, а також методи проєктування веборієнтованих інформаційних систем.

Магістерська робота містить аналіз предметної області пошуку клінічних досліджень, огляд сучасних мовних моделей і методів їх адаптації, розробку архітектури системи пошуку клінічних випробувань, реалізацію модуля обробки природномовних запитів та вебінтерфейсу користувача, а також експериментальне порівняння моделей Llama-3-8B, Gemma-3-27B та Qwen-3-4B у поєднанні з методами LoRA, QLoRA, DoRA та AdaLoRA.

Практичне значення роботи полягає у створенні прототипу інтелектуальної системи пошуку клінічних випробувань, яка спрощує доступ до медичної інформації, підвищує точність формування запитів до ClinicalTrials.gov та може бути використана для підтримки наукових досліджень і прийняття рішень у сфері медичної інформатики.

## ABSTRACT

Qualification work: 86 pages, 8 figures, 6 tables, 25 sources, 2appendices.

CLINICAL TRIALS, NATURAL LANGUAGE PROCESSING, LARGE LANGUAGE MODELS, PEFT, CLINICALTRIALS.GOV, WEB APPLICATION, MEDICAL INFORMATICS.

The object of the study is the process of searching and analyzing clinical trial data in open medical information systems.

The subject of the study is methods and software tools for transforming natural language queries into structured search parameters for clinical trials using large language models.

The goal of the work is to develop and study an intelligent clinical trial search system based on large language models with parametrically efficient retraining, as well as to evaluate the effectiveness of various PEFT architectures and methods for the task of transforming natural language medical queries.

The research methods include: system analysis, natural language processing methods, machine learning, parametrically efficient retraining of large language models, experimental comparative evaluation, as well as methods for designing web-oriented information systems.

The master's thesis contains an analysis of the subject area of clinical research search, an overview of modern language models and methods for their adaptation, the development of a clinical trial search system architecture, the implementation of a natural language query processing module and a web user interface, as well as an experimental comparison of the Llama-3-8B, Gemma-3-27B, and Qwen-3-4B models in combination with the LoRA, QLoRA, DoRA, and AdaLoRA methods.

The practical significance of the work lies in the creation of a prototype intelligent clinical trial search system that simplifies access to medical information, improves the accuracy of queries to ClinicalTrials.gov, and can be used to support scientific research and decision-making in the field of medical informatics.

CONTENT

# LIST OF ABBREVIATIONS, CONVENTIONS, SYMBOLS, UNITS AND TERMS

IS – information system

LLM – large language model

NLP – natural language processing

API – application programming interface

JSON – JavaScript Object Notation

HTTP – Hypertext Transfer Protocol

UI – user interface

SPA – single-page application

PEFT – parameter-efficient fine-tuning

LoRA – Low-Rank Adaptation

QLoRA – Quantized Low-Rank Adaptation

DoRA – Weight-Decomposed Low-Rank Adaptation

AdaLoRA – Adaptive Low-Rank Adaptation

GPU – graphics processing unit

VRAM – video random access memory

NCT – National Clinical Trial identifier

# INTRODUCTION

Rapid advancements in computer technologies, especially in artificial intelligence and natural language processing, are beginning to automate previously unautomatable information-processing tasks. Intelligent systems that analyze and organize large amounts of information are becoming of paramount importance, as they offer users easy and efficient access to relevant information. Efficient information processing is of immense importance in many fields, especially in medicine and clinical research.

Each year, clinical trials information is registered in thousands of new entries in open access databases such as ClinicalTrials.gov. However, users of these databases are often frustrated as search systems are poorly designed and require users specific to their fields and structures, making the information retrieval process inefficient for researchers, clinicians, and patients. Therefore, users are in great need of intelligent systems that understand natural language and provide relevant structured information on demand.

Overcoming this challenge becomes easier with the advancement of Large Language Models (LLMs). Large Language Models (LLMs) have the capabilities of contextual understanding, semantic identification, and deciphering complex natural-language sentence structures, enabling the translation of everyday, unstructured user queries into defined and reliable search terms. Recently, advances in Parameter-Efficient Fine-Tuning (PEFT) have allowed powerful LLMs to be adapted to specific tasks in an industry without the need for extensive training, resulting in the use of significant computational resources.

The object of this research is the development and optimization of a clinical-trial search system powered by large language models.

The subject of the research is the architectural characteristics and effectiveness of various LLMs (Llama-3-8B, Gemma-3-27B, Qwen3-4B) combined with PEFT techniques (LoRA, QLoRA, DoRA, AdaLoRA) for transforming natural-language queries into structured API filters.

The goal of this master's thesis is to develop a high-accuracy and efficient natural-language query processing module for a clinical-trial search system, conduct a comparative analysis of different approaches to LLM fine-tuning, and integrate the resulting module into a fully functional web application.

# 1 ANALYSIS OF THE SUBJECT AREA AND FORMULATION OF THE RESEARCH PROBLEM

## 1.1 Analysis of the subject area

### 1.1.1 Relevance of the task

Natural language processing (NLP) is a branch of artificial intelligence that enables computers to understand, interpret, and generate human language. While traditional AI allows computers to process structured data, NLP enables them to “understand” the semantic meaning of texts and statements.

The topic of using large language models (LLMs) to search for medical information is extremely relevant and important in today's world. The rapid development of this field has a significant impact on many aspects of healthcare and shows great potential for scientific progress.

In medicine, LLM-based NLP is widely used to analyze medical records, scientific publications, and clinical studies. For example, in clinical trial research, it helps researchers find relevant studies based on complex criteria. NLP technology can effectively assist in any medical task that requires a retrieval of specific information from vast databases.

NLP’s application in medical research would allow to minimize time spent retrieval of relevant clinical studies and quickly provide researchers and physicians access to the latest scientific information.

Today, NLP has evolved to the point where it can cover a wide range of tasks and assist researchers in many types of scientific activities. In turn, these developments have brought a number of benefits that an increasing number of professionals can enjoy.

In the field of bioinformatics, clinical trial search engines are becoming the backbone of scientific research, helping researchers find relevant clinical trials based on complex criteria. Leading medical institutions are actively using NLP to improve the efficiency of scientific work. Examples include:

- ClinicalTrials.gov - the world's largest database of clinical trials;
- OpenTrials API - open access platform for clinical trial data;
- PubMed - database of medical and biological publications.

These NLP applications advance intelligent search technology and improve the efficiency of scientific research.

In the pharmaceutical industry, neural network-based systems enable more effective analysis of clinical trial results and optimization of the drug development process. In scientific research, they facilitate systematic literature reviews and meta-analyses.

The use of large language models in clinical research search is a key element in the development of intelligent medical systems and artificial intelligence. Active interest in this topic in the modern world is reflected in the continuous growth of research and development, which demonstrates the importance of this area for further technological progress and improving the quality of medical care.

NLP requires large amounts of data for training. Models analyze text data over and over again until they learn to recognize semantic relationships and, ultimately, can accurately interpret complex queries. For example, to teach a model to recognize inclusion criteria for clinical trials, it needs to be provided with a huge number of query examples and corresponding structured parameters so that it can recognize semantic differences and accurately transform a natural language query into a structured format.

### 1.1.2 Issues in searching clinical trial databases

Modern clinical trial search engines, such as ClinicalTrials.gov, face a number of fundamental limitations that significantly complicate the search process for end users. The main problem lies in the complexity of forming structured queries, as users must have specialized knowledge of the database structure and filter syntax.

Limited Systems show very faint understanding of the meaning of the queries they are processing. Such systems are unable to capture synonyms, contextual relationships, and naturally spoken phrases highlighting the system deficiencies. Thus, searching for something in unlike wording generates rather distinct findings.

Also, without a genuine natural language interface, users are compelled to change their way of thinking to align with the system's logic, instead of the system adjusting to their way of thinking. Merging multiple search criteria together is particularly challenging as the user has to take into consideration filters for the study phase, the demographics, the geography, and the medical conditions at the same time.

These system limitations significantly slow down the search process, reduce the efficiency of researchers and physicians, and can sometimes lead to relevant clinical studies being overlooked due to a mismatch between the user's natural thinking and the structured logic of the database.

### 1.1.3 Overview of existing clinical trial search systems

With over half a million trials cataloged in over two hundred forty countries around the world, the US National Institutes of Health's ClinicalTrials.gov is a top solution despite its complexities. The program's features include advanced search with the ability to filter by study phase, recruitment status, regards to participants, area of the world, and more. Yet, the system still has a major disadvantage and that is the complicated nature of the queries and lack of user understanding required in order to efficiently utilize the site.

OpenTrials API represents an alternative approach, offering an open application programming interface for accessing aggregated clinical trial data. This platform integrates information from various sources, including ClinicalTrials.gov, the EU Clinical Trials Register, and PubMed. The advantage of OpenTrials is its unified data structure and programmatic access, which opens up opportunities for the development of specialized applications. However, the system is limited in its semantic search capabilities and does not support natural language queries.

OpenTrials API is yet another approach with an open API that allows access to aggregated clinical trial data. This service pulls data from ClinicalTrials.gov, EU Clinical Trials Register, and PubMed. One of the advantages of OpenTrials is data uniformity and access programmatically, as it allows the development of dedicated

applications. On the other hand, the system lacks semantic search and natural language search.

PubMed and PubMed Central being part of MEDLINE, provide access to searchable clinical research publications. Their searches use keyword, MeSH term, and abstract relevance to surface appropriate studies. Some other targeted enterprise tools, like the Cochrane Library, are limited to systematic reviews and meta-analyses concentrating on high-quality evidence-based data. They, however, lack direct access to unstructured information concerning the parameters of studies.

Commercial solutions, such as TrialsTracker and IBM Watson for Clinical Trial Matching, show the use of technology for improving search options. TrialsTracker, developed at Oxford University, specializes in the monitoring of completed but unpublished studies by means of automated tracking algorithms. IBM Watson uses artificial intelligence to match patients at clinical trials to patients with given medical attributes. Although innovative, these systems come with high costs and complexities of integration.

By examining the functional types of systems available, certain areas of overlap can be observed. The majority of systems require users to have knowledge of certain sets of terminology and data frameworks which create significant roadblocks for researchers without a technical background. The absence of advanced interfaces that can understand and process natural language severely hampers the efficiency of a search. Furthermore, available systems tend to ignore the interrelationships between medical concepts, which often results in incomplete and inaccurate output.

The problem of integrating data from different sources requires special attention. Each platform uses its own information encoding standards, which complicates their consolidation and comparative analysis. The lack of a unified semantic layer limits the possibilities for cross-platform search and analytics.

The conclusions of this analysis confirm the need to develop a new system that combines the advantages of existing solutions — the structured data of ClinicalTrials.gov, the openness of OpenTrials API, and the intelligence of modern NLP technologies. The approach proposed in this work to create a natural language

interface using finely tuned LLMs is aimed at overcoming the limitations described and ensuring a new level of accessibility of information about clinical trials.

### 1.1.4 Large language models in medical informatics

Large language models (LLMs) are changing how complex text data in health care is processed and analyzed. One such field is medical informatics, which is benefiting from LLM's understanding of context, semantic relationships, and specialized vocabulary. This is essential for working with clinical data.

Medical information is unique and requires LLMs to have more than an understanding of language. Models such as BioBERT, Clinical BERT and PubMedBERT have a better understanding of medical concepts, terminology, and context as they have been pretrained on medical text corpora, allowing them to effectively work with literature, medical documents, and clinical trials. These models demonstrate high accuracy in tasks such as medical entity name recognition, diagnosis classification, and information extraction from electronic medical records.

In the context of clinical trial search, LLMs open up new perspectives for the creation of intelligent interfaces. They allow natural language queries such as “early phase melanoma studies for adults in Europe” to be transformed into structured search parameters, taking into account synonymy, context, and implicit criteria. This capability is particularly valuable for clinicians and researchers who often lack the technical training to formulate complex structured queries.

An important aspect of LLM application in medical informatics is the processing of multi-component queries that include several criteria simultaneously: medical conditions, demographic parameters, geographic restrictions, and study design characteristics. Traditional systems are often unable to effectively process such complex queries, whereas LLMs can identify relationships between different components and interpret them correctly.

However, there are substantial hurdles when it comes to LLM and ML tech in the healthcare domain. Hallucination, and the generation of erroneous information, is exacerbated in the healthcare ecosystem as it possesses the potential to cause harm.

The risks associated with it can be alleviated through the meticulous model refinement with reputable medical information, as well as the implementation of result validation mechanisms.

Another challenge is the constraint imposed by the LLM's context window. This can lead to complications when analysing lengthy medical texts, or when answering complex questions. There are several techniques to mitigate this. Hierarchical document processing, as well as information extraction prior to full analysis, are frequently employed.

To adapt general LLMs to the healthcare ecosystem, the idiosyncratic nature of medical vocabulary and its rapid evolution should be taken into account. Medical terminology is dense, highly specialized, and it is replete with abbreviations and acronyms. With regards to this, it is imperative to employ crafted methodologies to pre-train, as well as fine-tune, the model.A promising direction is the combination of LLM with ontologies and knowledge graphs in the medical field, such as SNOMED CT, MeSH, or UMLS[29]. This combination allows for a better understanding of the semantic relationships between medical concepts and ensures a more accurate interpretation of queries.

In the context of clinical trials, LLMs can greatly simplify the process of selecting trials for patients, automate the extraction of inclusion/exclusion criteria, and facilitate systematic reviews. This opens up new opportunities for personalized medicine and accelerated scientific research.

The implementation of LLMs in medical informatics requires careful testing, validation, and adherence to security principles, but their potential to transform access to clinical trial information is enormous and promises to set a new standard for intelligent search systems in medicine.

## 1.2 Analysis of the technologies under study in existing systems

### 1.2.1 Analysis of the ClinicalTrials.gov and OpenTrials APIs

ClinicalTrials.gov API is the primary software interface for accessing the world's largest database of clinical trials. API version 2, which was used in this study,

offers a RESTful architecture with support for JSON and XML formats. The system provides advanced search capabilities through parameterized HTTP requests, allowing studies to be filtered precisely according to numerous criteria.

Key search parameters include:

- condition/disease - medical condition or disease;
- intervention/treatment - type of intervention or treatment;
- location - geographical location of the study;
- phase - phase of the clinical trial;
- study type - type of study design;
- eligibility criteria - criteria for selecting participants.

Technical features of the API include a pagination system for large result sets, rate limiting, and support for specific syntax for complex filters. For example, the filter by study phase requires the exact format “PHASE1”, “PHASE2”, etc., which creates additional challenges for the automatic transformation of natural language queries.

The OpenTrials API represents an alternative approach, aggregating data from various sources, including ClinicalTrials.gov, the EU Clinical Trials Register, and other public databases. The main advantage of OpenTrials is its unified data structure, which normalizes information from different sources, simplifying its further processing.

A comparative analysis of the two APIs reveals significant differences:

Table 1.1 – Comparative analysis of ClinicalTrials.gov API and OpenTrials API

| Characteristics | ClinicalTrials.gov API | OpenTrials API |
|---|---|---|
| Data structure | Rigidly defined, detailed | Unified, aggregated |
| Data relevance | Direct updates | Depends on sources |
| Query restrictions | Strict restrictions | More flexible |
| Documentation | Simple but less detailed | Detailed but complex Simple but less detailed |
| Filter support | Extensive | Limited |

For the purposes of the study, ClinicalTrials.gov API was chosen as the primary data source due to its completeness, relevance, and direct integration with the primary source. However, for further expansion of the system, a modular architecture is planned, which will allow OpenTrials API to be integrated as an additional source.

An important technical aspect is the processing of the ClinicalTrials.gov API response structure, which contains nested JSON objects with detailed information about the study, including:

- Information about sponsors and researchers;
- Basic metadata (NCT ID, title, description);
- Detailed participant selection criteria;
- Contact information and locations;
- Study results (if available).

Analysis of API performance revealed the need to implement caching and query optimization mechanisms to ensure acceptable system response times. Particular attention was paid to error handling and recovery from failures, which is critical for reliable system operation in a real-world environment.

Integration with the ClinicalTrials.gov API also required careful design of the input parameter validation system, as the API is sensitive to data format and has specific requirements for character encoding and value formatting.

### 1.2.2 Natural language processing methods for medical data

Due to the particularities of the medical field, the field of natural language processing will differ from all other NLP fields in his work of high importance, high specialized accuracy, and the uniqueness of medical language. We can characterize medical texts by specialized vocab, abbreviations, synonymy and complex syntactic structures, which leads to the need for specialized processing techniques.

A particular task in medical NLP, and one of the most impactful, is Clinical Named Entity Recognition (CNER). This technology is the ability to automatically extract and classify medical concepts in texts such as diagnoses, medications, procedures, anatomical entities, and biomarkers. For this task, classical approaches like CRF and SVM, and modern transformer-based architectures are used, which are fine-tuned on medical corpora.

The normalization of medical vocab is especially important because of the need for interoperability across the data silos. This process consists of mapping variant medical terms to standard concepts in coding systems such as:

- SNOMED C - comprehensive clinical terminology;
- MeSH - medical subject headings;
- ICD-10/11 - international classification of diseases;
- UMLS - unified medical language.

Understanding medical relations helps identify semantic relations such as treatment,"diagnosis," and ``side effect." This is crucial for automating clinical trials as one must identify relations between the interventions, diseases, and outcomes.

Current medical NLP practices incorporate the use of transfer learning and domain-specific linguistic models. The most notable cases involve:

– BioBERT - a BERT model trained on biomedical texts from PubMed and PMC;
– ClinicalBERT- specialized for clinical texts from electronic medical records;
– PubMedBERT - a model trained exclusively on texts from PubMed.

Semantic parsing and intent analysis methods are used to process natural language queries in the context of clinical research searches. These technologies allow you to determine the user's intentions and extract key search parameters from free text queries.

A distinctive feature of medical NLP is the need to take into account the contextual dependence of medical terms. For example, the term “cancer” can mean both an oncological disease and a zodiac sign, depending on the context. To solve this problem, methods of contextual disambiguation and multi-task learning are used.

The processing of elliptical and non-standard constructions in medical queries is another important aspect. Users often form queries with missing components or use colloquial expressions, which requires the implementation of semantic completion and normalization mechanisms.

Integrating ontologies and knowledge graphs with NLP enhances the comprehension of the semantic association between medical concepts and assures a sophisticated understanding of the semantics of the queries.

Expert-derived data, algorithm validation and ongoing evaluation of system performance are utilized to maintain quality and safety of medical NLP technologies. This is critical in the case of clinical practices, since mistakes could be detrimental to the patient.

The development of innovative technologies for natural language processing in the medical field is driven by the latest advancements in AI, together with a deep understanding of the medical domain.

### 1.2.3 Parametrically Effective Fine Tuning (PEFT)

Parameter-Efficient Fine-Tuning (PEFT) is currently considered one of the most promising approaches for adapting large language models (LLMs) to specific tasks. Its basic concept is to abandon full-scale retraining of all millions or billions of model parameters in favor of selective tuning of only a small portion of them. This methodology has become critical in resource-intensive fields such as medical data processing, where traditional fine-tuning is often impractical due to limitations in computing power and memory.

As part of this work, a number of key PEFT methods were investigated. Among them is LoRA (Low-Rank Adaptation), which is based on the hypothesis of the low-dimensional nature of changes during adaptation. It introduces low-dimensional decomposition matrices to approximate update gradients, which reduces the number of tunable parameters by several orders of magnitude. For models with billions of parameters, LoRA typically uses a rank of 4-32, reducing the number of parameters to tune to just a few million.

The QLoRA (Quantized LoRA) method is a natural evolution of LoRA ideas, combining low-dimensional adaptation with a 4-bit weight quantization technique[15]. This approach not only reduces the number of parameters to be tuned, but also dramatically reduces RAM requirements during training. QLoRA uses specialized NF4 (NormalFloat 4-bit) quantization and double quantization to optimize GPU memory usage, making it possible to train extremely large models on much more modest hardware.

A more innovative approach is DoRA (Weight-Decomposed Low-Rank Adaptation), which divides the learning process into two independent components: magnitude and direction. This decomposition provides a more accurate and controlled adaptation of the model to the target task, while retaining all the advantages of low-rank adaptation. The experimental results obtained in the study demonstrate that DoRA often outperforms standard LoRA in tasks requiring high accuracy.

Another advanced method is AdaLoRA, which introduces the concept of adaptive rank allocation for different parts of the model. Instead of a fixed distribution, this method dynamically evaluates the importance of different components (e.g., individual attentions or layers) for the target task and allocates computational resources according to this importance. This provides an optimal balance between efficiency and performance, allowing the model to focus on the most informative parameters.

The advantages of PEFT methods are particularly important for medical applications. The most obvious is the ability to reduce memory requirements, allowing powerful billion-parameter models to be used on standard hardware. This is directly related to the acceleration of the learning process, since only a small fraction of the parameters are updated. In addition, it becomes possible to effectively train several expert models for different medical specialties (e.g., oncology, cardiology) based on a single base LLM. It is also critically important to preserve the general knowledge of the model that was acquired during pre-training, which prevents the phenomenon of "catastrophic forgetting" when adapting to the medical domain.

In the context of the specific task of this work — processing clinical queries — PEFT methods demonstrate high efficiency. They allow for the precise adaptation of general LLMs to understand complex medical terminology, patient inclusion and exclusion criteria, and the specifics of the structure of clinical studies. As a result, high accuracy is achieved in transforming natural language queries from doctors into structured search parameters in medical databases, all with minimal computational overhead.

The experimental part of the work confirmed the high practical value of PEFT for parsing clinical queries. The best result was achieved using a combination of the Gemma-3-27B model and the DoRA method, which demonstrated an accuracy of 93.8%. This result is particularly significant because it was obtained with significantly reduced training costs compared to full fine-tuning, confirming the effectiveness of the chosen approach.

### 1.2.4 Comparison of LLM architectures for specialized tasks

In the context of medical information systems, the ability of a model to accurately interpret and structure specialized terminology is of particular importance. That is why, when comparing the selected architectures, their competence in recognizing medical terms, including variant names, common abbreviations, and context-dependent concepts, was analyzed first. In this regard, the Google/Gemma-3-27B model (27 billion parameters) showed the best results: a large number of parameters and a focus on scientific corpora enable it to interpret terms more confidently, which usually pose difficulties for less capacious architectures.

The model's ability to process complex, multi-component queries is the next key criterion. In clinical trials, queries often include information about the trial phase, type of pathology, patient demographics, geographic restrictions, time frames, and other filters. Architectures with a larger number of parameters, such as Gemma-3-27B, have an advantage in the semantic decomposition of such queries, as they find it easier to retain and reconcile extensive context, which directly improves the quality of formalized structures for searching clinical trial databases. Llama-3-8B (8 billion parameters) also demonstrates balanced performance: it has sufficient capacity to generate structured responses correctly while remaining moderate in terms of hardware requirements, making it an attractive compromise for systems that need to combine quality with practical operation on standard equipment. The more compact Qwen/Qwen3-4B-Instruct-2507 (4 billion parameters) offers the lowest computing resource requirements while maintaining competitive quality in tasks where response speed and the ability to perform mass parallel processing are critical.

In practical implementations, it is essential to consider computational efficiency and performance: the model must comply with the resource constraints of the deployment environment. Qwen3-4B, being the smallest of those considered, proves to be extremely useful for prototypes and systems that process a large flow of simple or moderately complex queries in real time, while Llama-3-8B offers a

convenient middle ground for cases where a balance between performance and quality is needed.

The adaptability of the models is another important aspect. All three architectures (Llama-3-8B, Gemma-3-27B, and Qwen3-4B) can be effectively fine-tuned on medical corpora, but Gemma-3-27B, thanks to its scalability, provides a particularly noticeable increase in accuracy when using modern methods of effective retraining. In particular, the combination of PEFT (parameter-efficient fine-tuning) and DoRA techniques yields the best results for Gemma-3-27B, making this model the optimal choice for scenarios where semantic accuracy and in-depth terminology work are of utmost importance.

Therefore, analysis of critical factors—including accuracy of term interpretation, ability to handle multifaceted queries, resource requirements, and adaptability—indicates that the final choice of architecture should be based on specific system requirements. Gemma-3-27B (27B) is best suited for high-precision medical applications, Llama-3-8B (8B) for tasks requiring a balance between quality and efficiency, and Qwen3-4B (4B) for lightweight deployments and rapid prototyping.

## 1.3 Formulation of the research problem

Based on the analysis of the subject area and existing systems, the main research objective was formulated, which is to develop a highly efficient clinical research search system with a natural language interface. This system should ensure accurate transformation of user queries into structured API parameters through the use of large language models and parametrically efficient fine-tuning methods.

This system should ensure accurate transformation of user queries into structured API parameters through the use of large language models (LLM) and parametrically efficient fine-tuning (PEFT) methods.

The main goal of the work is to create an intelligent search engine that significantly simplifies access to information about clinical trials for researchers, doctors, and patients.

To achieve the set goals, the following set of interrelated tasks must be solved:

1. Theoretical Tasks

– Theoretical Tasks Analysis of the subject area and solutions: Study of existing approaches to processing medical texts and searching clinical research databases. Study of LLM models: Research on the architectures of large language models and methods for adapting them to specialized tasks (PEFT).

– Formalization and Modeling: Develop a formal mathematical model for transforming natural language queries into structured API parameters, including the formulation of the query parsing problem and Intent Recognition.

– Justification of choice: Justification of the choice of LLM architecture and PEFT method to achieve a target accuracy of 93.8% (Gemma-3-27B + DoRA))

2. Practical Tasks

– Dataset formation: Creation of a training dataset that combines synthetic natural language queries with real clinical trial metadata.

– Implementation and fine-tuning: Implementation and fine-tuning of three LLM architectures (Llama-3-8B, Gemma-3-27B, Qwen3-4B) using four PEFT methods (LoRA, QLORA, DORA, AdaLoRA).

– Web application development: Creation of a web interface (e.g., using Angular) and backend (e.g., on FastAPI).

– API integration: Integration of the system with the ClinicalTrials.gov API and implementation of caching mechanisms to optimize speed.

3. Experimental Tasks

– Accuracy Testing: Comprehensive testing of query transformation accuracy

– Performance Evaluation: Evaluation of system performance under various load conditions

– Usability Testing: Testing the usability of the interface (System Usability Scale score.

– Comparative Analysis: Conducting a comparative analysis with existing search engines to confirm the advantages of the developed solution.

# 2 MATHEMATICAL BASIS AND SYSTEM ARCHITECTURE

## 2.1 Mathematical model of natural language query transformation

### 2.1.1 Formal formulation of the query parsing problem

The task of parsing natural language queries into structured search parameters can be formalized as a mapping from the space of natural language queries to the space of structured API parameters. Let Q be the space of natural language queries, where each queryq ∈ Q represents a sequence of words $q = (w_1, w_2, ..., w_n)$. Let be a space of structured parameters, where each parameter p ∈ P is a tuple $p = (param_1, param_2, ..., param_k)$. Then the problem is to find the optimal mapping f: Q → P, that minimizes the loss function $L(f(q), p_(gt))$, де $p_(gt)$ is the ground truth of the parameter values.

For the ClinicalTrials.gov API, the output parameters have a clearly defined structure:

p = {

"query.cond": "condition/disease (medical condition or disease)

"filter.advanced": "Essie expression", (advanced filters in Essie syntax)

"query.locn": "location", ( geographic location)

"query.intr": "intervention", (type of intervention)

"query.phase": "phase", (study phase)

"query.age": "age range", (age restrictions)

"query.gender": "gender" (gender criteria)

}

Each parameter is defined from corresponding sets of values. For example, medical conditions c ∈ C may belong to MeSH terms, ICD-10 codes, or colloquial names of diseases. Filters use Essie syntax to combine conditions, for example: "AREA[Phase]PHASE3 AND AREA[MinimumAge]18 YEARS".

Parsing quality is evaluated using accuracy metrics. Exact Match Accuracy is defined as $EM = \frac{1}{N} \times \sum_{i=1}^{N} \mathbb{1}[f(q_i) = p_i^{(gt)}]$, where $\mathbb{1}$ - is an indicator function that returns 1, if the predicted parameters exactly match the ground truth, and 0 otherwise. Component Accuracy evaluates individual components of the parameters: $CA_j = \frac{1}{N} \times \sum_{i=1}^{N} \mathbb{1}[f_j(q_i) = p_i^{(gt)}]$. Generalized accuracy uses the Jacquerot measure: $CA_j = \frac{1}{N} \times \sum_{i=1}^{N} \frac{\left[f(q_i) \cap p_i^{(gt)}\right]}{\left[f(q_i) \cup p_i^{(gt)}\right]}$.

The task has semantic equivalence constraints: ∀q ∈ Q: Semantic(f(q)) ≡ Semantic(q), which guarantees that the content of the query is preserved after transformation. Syntactic correctness is also required: ∀p ∈ P: ValidAPI(p) = true, which ensures the correctness of the generated parameters for the API.

Queries are classified by complexity as simple (one condition), medium (2-3 conditions), and complex (4+ conditions with logical operators). To process ambiguous queries, a probabilistic model is used, where the optimal parameters are found as $p^* = \text{argmax}_{p \in P} P(p|q)$, using Bayes' theorem to calculate the posterior probability.

The model training process is formalized as minimizing the loss function: $\min_{(\theta)} \Sigma_{(q,p)} \in D\ L(f\theta(q), p) + \lambda R(\theta)$, where θ are model parameters, D is a training dataset of “query-parameter” pairs, L is a loss function (e.g., cross-entropy), R(θ) is regularization, and λ is the regularization coefficient. This formalization provides a clear mathematical basis for developing and evaluating algorithms for parsing natural language queries in the context of clinical research search.

### 2.1.2 Intent Recognition

The intent recognition model is a critical component of natural language query processing systems, as it allows the identification of the user's primary goal and the corresponding type of search operation. In the context of clinical trial searches, user intent can range from simply obtaining general information to specific queries regarding inclusion/exclusion criteria.

Formally, the task of determining intent can be represented as a mapping from the query space to the discrete intent space: $g: Q \rightarrow I$, where $I = \{i_1, i_2, ..., i_m\}$ - is a

finite set of possible intents. The following main categories of intent have been defined for the clinical research search system:

I = {“search_by_disease”,“filter_by_phase,“geographic_search,“demographic_search, “combined_search, “information_request”}

Each intention i ∈ I has a corresponding probabilistic model P(i|q), which is calculated based on the semantic and syntactic features of the query. A Bayesian approach is used to classify intentions: $i^* = \text{argmax}_i \in I\ P(i|q) = \text{argmax}_i \in I\ P(q|i)P(i)/P(q)$. The probability P(q|i) is modeled using large language models that take into account the context and semantics of the query.

To improve the accuracy of intent determination, contextual features are used, including keywords, syntactic constructions, and semantic patterns. For example, queries with the keywords “phase,” “stage,” or ‘фаза’ are highly likely to correspond to the intent “phase_filtering,” while geographic names indicate “geographic_search.”

The intent determination model is integrated with the overall system architecture and interacts with the parameter parsing module. After identifying the intent, the system applies specialized algorithms to extract the relevant parameters. For example, for the intent “combined_search,” mechanisms for processing complex logical expressions are activated, while for “information_request,” a simplified response model can be used.

The effectiveness of the intent prediction model is evaluated using classification accuracy metrics: Accuracy = (TP + TN)/(TP + TN + FP + FN), де TP, TN, FP, FN are the number of true positives, true negatives, false positives, and false negatives, respectively. The $F_1$-score is also used to balance precision and recall.

Implementing an accurate intent model significantly improves the user experience by providing relevant search results and reducing the number of unsuccessful queries. This is particularly important in the medical field, where the accuracy of information can be critical to clinical decision-making.

## 2.2 Transformer Architecture: Principles of Construction and Attention Mechanisms for Medical Text Processing

Transformer architecture has become the fundamental basis for modern large language models. Unlike previous architectures, such as recurrent neural networks (RNNs) and convolutional neural networks (CNNs), transformers rely solely on attention mechanisms to process sequences, allowing them to process long text sequences and capture complex semantic dependencies efficiently [6].

The overall architecture of the Transformer, which includes a multi-layer encoder and decoder, can be seen in Figure 1.

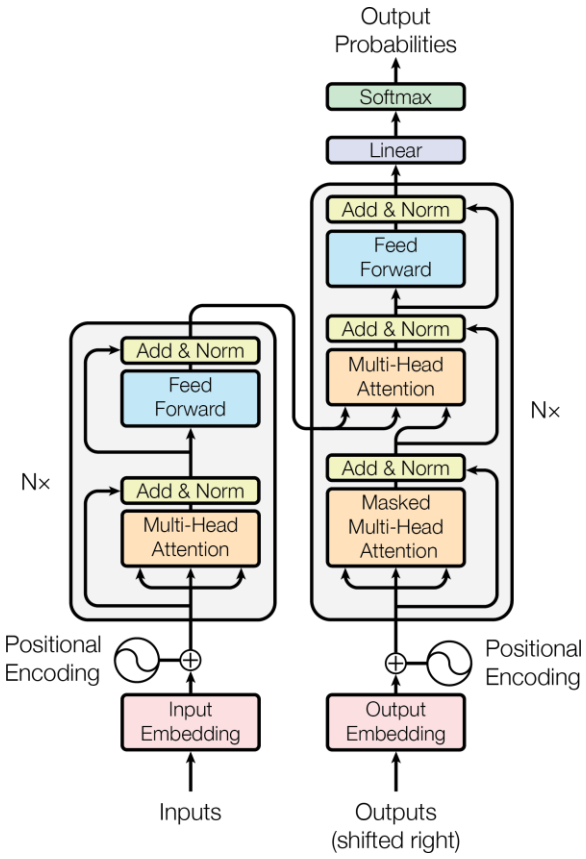



Figure 1: The Transformer - model architecture[6]

Key components of transformer architecture:

– The Scaled Dot-Product Attention is the main innovative component of transformers. This mechanism allows the model to determine the importance of each word in a sequence relative to all other words. Mathematically, self-attention is calculated using the formula: $Attention(Q, K, V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$ , where Q (query), K (key), V (value) - are matrices obtained by linear transformations of input embeddings, $QK^T$ is a matrix of attention scores (logits) that shows how much each

element in query Q corresponds to each element in key K, a $d_k$ is the dimension of the keys;

– Multi-Head Attention extends the self-attention mechanism, allowing the model to simultaneously focus on different aspects of information from different representative subspaces. This is achieved by parallel computing of multiple attention mechanisms and then combining their results;

– Positional encoding provides the model with information about the order of words in a sequence, since transformers, unlike RNNs, do not have a built-in mechanism for accounting for sequential order. The positional encoding vector PE for position pos (starting from 0) and embedding dimension dmodel is calculated as follows:

For even dimension indices i (where i = 0, 2, 4, …): $PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i}/d_{model}}\right)$

For odd dimension indices i (where i = 1, 3, 5, …):

$PE_{(pos,2i)} = \cos\left(\frac{pos}{10000^{2i}/d_{model}}\right)$;

– The encoder layers consist of two main components: Multi-Head Self-Attention and Position-wise Feed-Forward Network (FFN). Each layer also contains residual connections and layer normalization, which contributes to stable training of deep architectures.

The basis of the attention mechanism is a weighted summation operation, where each element of the sequence is assigned a weight according to its relevance: $a_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)}$, wherere $e_i$ is assessment of the similarity (logit) between the query and the key, calculated using a similarity function. The evolution of attention mechanisms has led to several key forms:

– Additive Attention [8] used a nonlinear function to calculate the estimate: $e_i = v^T \tanh(W_1 h_i + W_2 s)$;

– Dot-Product Attention [9] simplified the calculation using the scalar product: $e_i = h_i^T s$.

Scaled Dot-Product Attention described above, is the most efficient version for Transformer, providing high parallelization [6].

In the context of large language models, decoder-only architectures optimized for text generation tasks are used. These models typically consist of several dozen or even hundreds of transformer blocks, each containing attention mechanisms and feed-forward layers.

Advantages of transformer architecture for processing medical texts:

– parallelization of computations: the ability to process the entire sequence simultaneously significantly speeds up training and inference;

– effective modeling of long dependencies: the ability to capture relationships between words separated by a significant distance in the text;

– Scalability: the architecture allows for effective increases in model size and training data volume;

– Adaptability to different tasks: the ability to fine-tune for specific domains, such as medicine.

For medical applications, transformer architecture is particularly valuable due to its ability to process complex medical terms, recognize contextual relationships between symptoms, diagnoses, and treatments, and effectively interpret multi-component queries about clinical studies.

Transformer architecture has found widespread use in leading medical information systems and research platforms: IBM Watson Health, Google Health's BERT-based models, Amazon Comprehend Medical, and Microsoft's BioGPT.

In the context of clinical trial search, transformer architecture enables systems such as ClinicalTrials.gov and OpenTrials to understand semantically complex queries such as “early-phase immunotherapy trials for melanoma in adult patients with elevated PD-L1” by transforming them into precise, structured search parameters.

## 2.3 Methods of parametrically efficient tuning

### 2.3.1 Mathematical foundations of the LoRA method

The Low-Rank Adaptation (LoRA) method is based on the hypothesis that weight changes (ΔW) during fine-tuning are low-rank in nature (r) [10]. This allows the model to be adapted efficiently by freezing the initial weights $W_0$ and training only a small number of new parameters.

Let's start with low-rank decomposition. For a given weight matrix $W \in R^{d\times k}$ the update ΔW is represented as a decomposition into two low-dimensional matrices: $\Delta W = BA$, where $B \in \mathbb{R}^{d\times r}, A \in R^{r\times k}$, adaptation rank $r \ll \min(d, k)$.

Forward propagation through a layer with adapted weights W + ΔW looks like: $h = W_0 x + \Delta W_x = W_0 x + BA_x$, where $W_0$ are the frozen initial weights. Only the parameters B and A are learned, which radically reduces the number of trained parameters.

Minimization of the loss function $\mathcal{L}(\theta)$ is achieved by calculating the cross-entropy loss $\mathcal{L}_{CE}$ at the output of the model $f$: $\mathcal{L}(\theta) = \mathcal{L}_{CE}(f(x;\ W_0 + \Delta W), y)$ . The scaling factor $\alpha$ is often used to normalize ΔW.

### 2.3.2 Training algorithm using DoRA (Weight-Decomposed Low-Rank Adaptation)

DoRA extends LoRA by decomposing weights into magnitude and direction[11]. This allows the model to be more accurately adapted to the specifics of medical data while preserving the general knowledge encoded in the original weight values. The initial weights W are decomposed into a vector of magnitudes m and a matrix of directions V according to the formula: $W = m \cdot \frac{V}{\|V\|_c}$, where m is the vector of magnitudes, V is the matrix of directions, and $\|\cdot\|_c$ is the column norm.

In DoRA, adaptation occurs both for the direction V using standard LoRA ($V_{adapted} = V_0 + BA$), and for the value m using the scalar Δm $m_{adapted} = m_0 + \Delta m$ .

During the training phase, the adapted weights $W_{adapted}$ are calculated as:

$$W_{adapted} = (m_o + \Delta m) \cdot \frac{V_0 + BA}{\|V_0 + BA\|_c}$$

Optimization is achieved by minimizing $\mathcal{L}_{CE}$ losses with respect to parameters B, A and $\Delta m$. To stabilize training, which is critical when working with sensitive medical data, gradient normalization is used, as well as adaptive training with different learning rates for components B, A and $\Delta m$.

The high reliability and accuracy of DoRA makes it the optimal choice for high-precision medical query parsing tasks where stability is critical.

## 2.4 System performance metrics

### 2.4.1 Exact Match Accuracy

Exact Match Accuracy (EM) is a fundamental metric for evaluating parsing quality. It measures the percentage of queries for which all components of the output JSON match the reference values exactly.

Mathematically, EM accuracy is calculated as the average value of the indicator function for N test requests: $EM = \frac{1}{N}\sum_{i=1}^{N} I(JSON_{pred}^{(i)} \equiv JSON_{true}^{(i)})$, where N — is the total number of test queries; I(·) is an indicator function that returns 1 if two JSON objects are completely identical; $JSON_{pred}^{(i)}$ та $JSON_{true}^{(i)}$ are the generated and reference JSON for the i-th query, respectively.

For clinical queries, the criteria for complete matching require all key fields to be identical, including query.cond, filter.advanced, query.locn, and all additional parameters (phase, age, gender, etc.). It is important to note that EM is a very strict metric: it does not take partial matches into account and is sensitive to minor differences in formatting or syntax, which may be too harsh for evaluating semantically equivalent results.

### 2.4.2 Component Accuracy

Component Accuracy (CA) provides a more detailed assessment of system performance by measuring the accuracy of individual components of a structured query, allowing you to identify weaknesses in the system. The mathematical definition of CA for a specific field (component) is calculated as:

$$CA_{component} = \frac{1}{N}\sum_{i=1}^{N} I(component_{pred}^{(i)} \equiv component_{true}^{(i)}).$$

In the context of parsing clinical queries, it is critical to evaluate the accuracy of the following components, in particular the accuracy of medical conditions ($CA_{cond}$), the accuracy of filters ($CA_{filter}$), and the accuracy of demographic parameters ($CA_{demo}$), which are specified in Table 2.1.

Table 2.1 - Key component accuracy (CA) metrics for evaluating the quality of clinical query parsing

| Component | Definition | Formula |
|---|---|---|
| Condition Accuracy | $CA_{cond}$ | $CA_{component} = \frac{1}{N}\sum_{i=1}^{N} I(query.cond_{pred}^{(i)} \equiv query.cond_{true}^{(i)})$ |
| Filter Accuracy | $CA_{filter}$ | $CA_{component} = \frac{1}{N}\sum_{i=1}^{N} I(filter.advanced_{pred}^{(i)} \equiv filter.advanced_{true}^{(i)})$ |
| Accuracy of Demographic Parameters | $CA_{demo}$ | $CA_{component} = \frac{1}{N}\sum_{i=1}^{N} I(demographic_{pred}^{(i)} \equiv demographic_{true}^{(i)})$ |

To improve the relevance of the evaluation, especially for medical conditions and filters, semantic equivalence is used instead of strict identity (e.g., “breast cancer” may be equivalent to “mammary carcinoma”). Finally, since different components have different importance for the final search, weighted component accuracy (WCA) is applied, which sums the accuracy of the components multiplied by their expert-determined weights $\omega_i$: $WCA = \sum_i \omega_i \cdot CA_i$.

This combination of metrics provides a comprehensive assessment of the system's quality, allowing for accurate measurement of both overall performance and the performance of individual components in the context of clinical query parsing.

# 3 METHODOLOGY AND EXPERIMENTAL ENVIRONMENT

## 3.1 Selection of models and tuning methods

### 3.1.1 Criteria for selecting LLM architectures

The selection of large language model architectures for the task of parsing clinical queries was based on a comprehensive evaluation across five key criteria that balance power, efficiency, and domain relevance. The first critical factor is the efficiency of medical terminology processing: the model must demonstrate a high level of understanding of specialized medical vocabulary, including synonym recognition (“carcinoma” and “cancer”), abbreviation processing (T2DM, HER2+), and context-dependent meaning differentiation. The second criterion is scalability and computational efficiency, which takes into account the number of parameters (optimally 4B–27B for a balance of power and accessibility), VRAM requirements, and real-time query processing speed. The third important aspect is the quality of instruction execution, which includes the ability to accurately execute complex JSON formatting instructions and correctly apply the specific syntax of the ClinicalTrials.gov API.

Also, the choice of LLM is justified by domain specialization, which includes prior training on biomedical texts (PubMed, medical literature) and the ability to reason medically. Finally, practical suitability for deployment is evaluated, including support for inference tools (Ollama, vLLM), availability of open model weights, and quality of documentation.

The Llama architecture [12] uses an optimized Transformer architecture (specifically, Grouped-query Attention and SwiGLU), which allows for higher quality at lower computational costs compared to other models of similar size. Gemma models [13] inherited key architectural innovations from Gemini research and were developed with a focus on responsible AI, which is critical for sensitive medical applications. These models, especially the versions with Instruct fine-tuning, demonstrate high accuracy in tasks that require logical reasoning. In addition, the Qwen architecture [14] was trained on a high-quality and diverse dataset, demonstrating strong multilingual

understanding capabilities that may be useful for parsing clinical queries in multinational studies.

Table 3.1 - Evaluation of selected LLM architectures

| Model | Size | Advantage | Disadvantage |
|---|---|---|---|
| Meta-Llama-3-8B-Instruct | 8B | Optimal balance of quality and efficiency, strong instructional capabilities | Limited medical specialization |
| Google/Gemma-3-27B-it | 27B | High accuracy due to scale, improved understanding of scientific texts | Higher VRAM and resource requirements |
| Qwen/Qwen3-4B-Instruct | 4B | Efficiency for limited environments, fast inference | Potentially lower accuracy for complex multi-component queries. |

### 3.1.2 Justification for the choice of PEFT methods

The choice of parametrically efficient tuning methods is justified by the specific requirements of the clinical query parsing task and the limitations of computational resources, in particular the need for training on a GPU with 24GB VRAM. The first critical requirement is memory efficiency, which includes the ability to train on available hardware and minimize memory requirements during inference. The second key requirement is the preservation of the model's general knowledge, which involves minimizing catastrophic forgetting and preserving the ability to understand general language after domain adaptation.

In addition, an important factor is the convergence speed on small domain-specific datasets and the flexibility and adaptability of the method, which allows for layer-by-layer tuning and adaptation to different types of medical queries. Previous studies have shown that for medical NLP tasks, where accuracy is critical, methods with extended adaptation mechanisms (DoRA, AdaLoRA) demonstrate an advantage over basic approaches[16].

## 3.2 Formation of a training dataset

### 3.2.1 Sources of clinical research data

Three independent sources were used to form the training dataset, ensuring diversity, clinical relevance, and structured information.

The main source of real data was the TrialGPT collection, which contains over 75,000 clinical trials obtained from ClinicalTrials.gov. This corpus has been further cleaned and normalized, allowing it to be used for training models without additional preprocessing.

The data includes:

- NCT identifiers, study titles, and descriptions;
- information about conditions and interventions;
- clinical trial phases (Phase 1–4);
- criteria for inclusion/exclusion of participants;
- age and gender restrictions;
- geographical locations, centers, and contact details;
- type of design (interventional/observational).

This set was used as a basis for forming structured JSON responses, which were matched with synthetic natural language queries.

The next source is CliniFact, a domain-oriented set of medical facts that contains structured clinical statements as well as relationships between medical conditions, interventions, and demographic parameters[17]. The corpus includes:

- normalized medical terms (MeSH, UMLS);
- statements about treatment, risk factors, medical events;
- relationships between diseases and therapies;
- medical categories and semantic roles.

CliniFact was used for semantic alignment of terminology and enhancement of synthetic query quality: language variants (“breast cancer,” “mammary carcinoma”), demographic concepts (“older adults” → ≥65), specific medical attributes.

The third source was synthetically generated natural language queries created based on ClinicalTrials.gov structures and taking into account the instructor's recommendations for covering various search scenarios. Synthetic examples filled in the gaps in real data and ensured variability in wording.

### 3.2.2 Generation of synthetic queries

Since there are no large corpora of natural language queries for ClinicalTrials.gov available in open sources, 500 synthetic queries, generated manually and automatically, were added to the training set. The purpose of generating synthetic data is to cover scenarios that are rarely encountered in real datasets, to account for different styles of natural language formulations, and to create realistic examples of structured JSON responses. Synthetic queries are divided into simple ones with 1-2 components, medium ones with 3 components, and complex ones with more than 4 components. There are also ambiguous formulations among synthetic queries, such as “postmenopausal women,” which expects a result of MinimumAge ≥ 45, Gender = Female, and other queries.

Synthetic queries were matched with structured JSON outputs according to the ClinicalTrials.gov API:

```
{
  "query.cond": "diabetes",
  "filter.advanced": "AREA[Phase]PHASE3 AND AREA[Location]United States",
  "query.age": "Adult",
  "query.gender": "All"
}
```

### 3.2.3 Annotation and validation of data

To ensure the quality of the dataset, a combined approach to annotation was used: automatic, semi-manual, and semantic-logical control.

In the initial step, the data underwent automatic processing, during which key medical terms were normalized, clinical trial phases were unified, and age and

geographic characteristics were converted to formats specified by ClinicalTrials.gov API. This stage allowed for the rapid preparation of a large volume of real records, but could not fully take into account the peculiarities of natural language semantics, so further manual verification was mandatory.

As part of the manual annotation, a detailed expert review of synthetic and some real examples was carried out. Particular attention was paid to the correspondence of the structured response to the original natural language query, as well as the correct interpretation of concepts such as “postmenopausal women,” “older adults,” “advanced breast cancer,” or geographical generalizations such as “Western Europe.” In case of discrepancies, the parameters were refined manually to avoid inconsistent or contradictory annotations. For example, in several cases, automatic processing models incorrectly interpreted age groups or combined mutually exclusive filters, which required human intervention.

An additional validation tool was an internal API tester that sent structured queries directly to the ClinicalTrials.gov API. This verified the syntactic validity of Essie filters, the compliance of parameter formats with specifications, and the ability of queries to return correct results. If the API rejected the query or returned an error, the example was reviewed and corrected. At this stage, a number of structures were filtered out in which the combination of parameters contradicted the logic of clinical trials, such as cases where age or phase filters technically matched the format but had no real meaning.

The final stage was a semantic check based on CliniFact and MeSH/UMLS data [18]. This was necessary to harmonize terminology and ensure the correct representation of synonymous or related concepts. Such verification made it possible, for example, to establish the equivalence between “mammary carcinoma” and “breast cancer,” or to clarify that “postmenopausal women” correlates with age and gender criteria in a structured response. This made it possible to avoid semantic conflicts and ensure data consistency, which is an important prerequisite for further model training.

As a result of the annotation, the dataset acquired the necessary internal consistency, semantic accuracy, and formal compliance with API specifications.

## 3.3 Experimental environment and tools

### 3.3.1 Hardware

The experimental part of the study was performed on a computing node of the university GPU cluster equipped with an NVIDIA RTX A5000 graphics processor with 24 GB of video memory[20]. This hardware limitation significantly influenced the choice of methods for fine-tuning language models, since full training of models with 8–27 billion parameters on such a configuration is impossible due to memory constraints[14, 20]. Under these conditions, the use of parametrically efficient methods, such as LoRA, QLoRA, DoRA, and AdaLoRA, became the only practical approach that allowed the models to be adapted without exceeding the available resources.

All experiments were conducted in a dedicated working environment with access to a GPU during training sessions. Training, inference, and dataset processing were performed locally on a single node, which ensured the reproducibility of the results. The training time for a single configuration (model × PEFT method) ranged from several tens of minutes to several hours, depending on the model size and the complexity of the adaptation matrices. GPU memory was also critical during the inference stage, as the Gemma-3-27B and Llama-3-8B models required optimized weight loading, and in the case of Gemma, the use of 4-bit quantization to fit the model into the available VRAM.

### 3.3.2 Software libraries and frameworks

The software environment for the experiments was built on a modern stack of libraries focused on training and inference of large language models. The PyTorch framework served as the basis, providing a computational foundation for working with tensor operations and offering reliable support for GPU computing [21]. On top of it, the Transformers module was used, which provided interfaces to the Llama, Gemma, and Qwen models, as well as allowed working with their instruction versions in a standardized format.

To implement parametrically efficient fine-tuning, the PEFT package was used, which supports the most common modern methods of adapting large language models, including LoRA, QLoRA, DoRA, and AdaLoRA. This allowed the library to provide a unified interface for experiments with different methods and apply the same hyperparameters, simplifying further comparison of results. Although QLoRA requires additional components for quantization, this functionality was implemented using the bitsandbytes library, which was used to work with 4-bit weights.

The inference process also utilized optimizations provided by the Accelerate framework, which allowed for efficient load balancing and control of GPU memory usage. Standard Python libraries such as pandas for working with tables and JSON structures, as well as tools for normalizing text fields, were used to prepare, clean, and pre-analyze the dataset.

### 3.4 Methodology for training and evaluating models

The experimental part of the study involved a coordinated procedure for training and subsequent evaluation of three language models: Llama-3-8B, Gemma-3-27B, and Qwen3-4B. The same hyperparameters and the same set of training data were used for all configurations, which allowed for a correct comparison of both architectures and parameter-efficient fine-tuning methods.

Each model was trained in PEFT mode using LoRA, QLoRA, DoRA, and AdaLoRA methods. This approach was due to hardware limitations and the need to work with large models on a GPU with 24 GB of memory, where full retraining is impossible. The same rank and scaling parameters were used for all methods, which minimized the impact of configuration differences on the final results. Training was conducted over a single epoch, as previous tests had shown that additional epochs did not provide significant improvement, but only increased the risk of overfitting on a relatively compact dataset.

After training, each model was evaluated on a separate test set that was not used during training. The main metric was Exact Match Accuracy, which measures the model's ability to generate a structured JSON query that fully matches the reference

query. This metric is strict but most relevant to the task of transforming clinical queries, as even a small syntactic shift or incorrect filter can make an API query incorrect.

For a deeper analysis, component accuracy was used, which separately evaluates the correctness of recognizing conditions, phases, demographic parameters, and complex Essie filters. This approach made it possible to identify the types of information in which models make the most errors, as well as to compare architectures in terms of their resistance to complex multi-component queries.

Special attention was paid to semantic verification of structured responses. Even with formal matches, logical errors sometimes occurred, such as incorrect combinations of age parameters or contradictory conditions in filters. To avoid such cases, the model output was additionally analyzed for medical content compliance, which allowed for a more objective assessment of the system's quality.

As a result, this methodology provided reproducible results and allowed for a correct comparative study of PEFT methods for the task of parsing clinical queries. The Gemma-3-27B configuration in combination with the DoRA method demonstrated the best performance, which is consistent with expectations regarding the impact of model size and weight distribution on adaptation quality.

# 4 DEVELOPMENT OF A CLINICAL TRIAL SEARCH SYSTEM

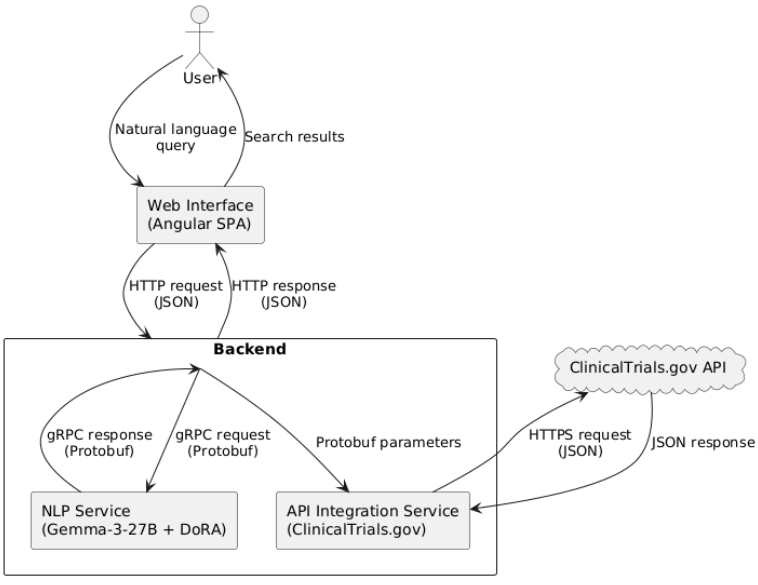
## 4.1 System architecture

The developed system is built on a modular principle and consists of three main components: a natural language query processing module, a ClinicalTrials.gov API integration module, and a web interface for end users. This distribution ensures the independence of the logical parts of the system, simplifies scaling, and allows individual components to be adapted without affecting others.

The key element is the server part, which accepts requests in any form and converts them into structured parameters that meet the requirements of ClinicalTrials.gov. After that, the data is obtained from an external API, filtered, and transferred to the web interface for display.

Internal interaction between modules is organized through Protocol Buffers (protobuf), which provides compact data serialization, clear contracts between services, and better performance compared to JSON [22].

The overall architecture of the system is shown in Figure 4.1.




Picture 4.1 – High-Level Architecture of the Clinical Trial Search System

### 4.1.1 Natural language query processing module

The natural language query processing module is the central part of the system, as it is responsible for the intelligent interpretation of user queries and the formation of

corresponding structured parameters. Its operation is based on the large language model Gemma-3-27B, which was adapted using the DoRA method. This configuration ensured the highest accuracy of semantic analysis, especially in cases of complex multi-component clinical queries.

Text processing takes place in several consecutive stages. First, the query undergoes light normalization, which removes unnecessary characters and aligns the sentence structure. The text is then sent to the language model, which uses an instructional prompt to generate a structured response in JSON format. The model highlights the main elements of the clinical context — diseases, interventions, study phase, geographic restrictions, and additional criteria. A separate result of the work is a field with an extended logical expression formed in the Essie syntax supported by ClinicalTrials.gov.

After generation, the structured object undergoes internal verification. Unified parameters are packaged into a protobuf message, which is transmitted to the API integration module. Using protobuf at this stage avoids formatting ambiguities and ensures stable inter-service interaction. At this stage, the values are analyzed for compliance with the expected formats: research phases must be marked as “PHASE1”, “PHASE2”, etc.; age and geographic parameters must have a unified form; Essie expressions are checked for correct construction. If JSON does not meet API requirements, the system automatically detects an error and prompts the user to clarify the request.

Thanks to the combination of a large model and DoRA adaptation, the module is able to correctly process free-form queries, including terms that often have contextual restrictions, such as “postmenopausal women,” “early-stage lung cancer,” or “trials in Western Europe.”

To illustrate the end-to-end processing pipeline of a natural language clinical query, Figure 4.2.1 illustrates the flow of natural language query processing and parameter extraction, while Figure 4.2.2 presents the interaction with the ClinicalTrials.gov API and the processing of search results.

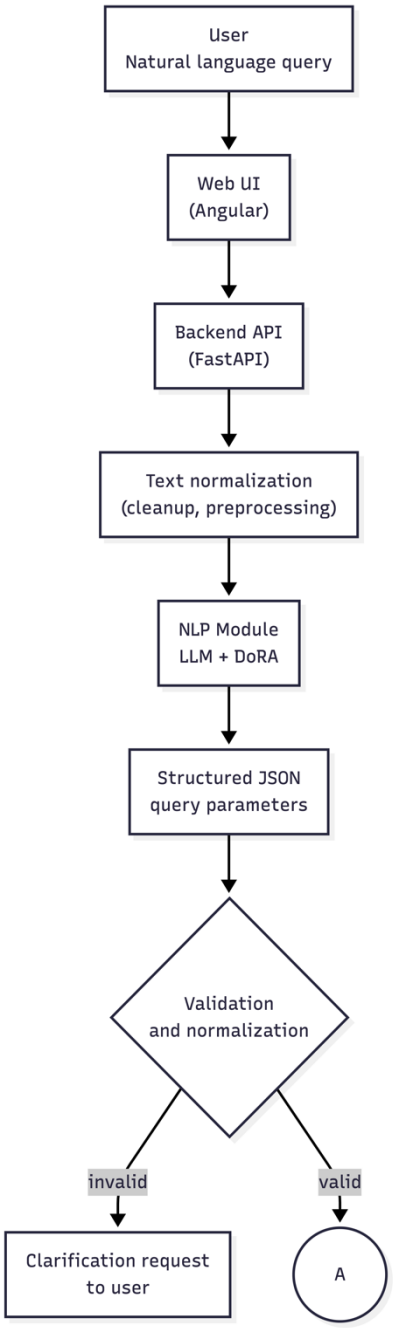



Figure 4.2.1 — Flow diagram of natural language query processing and parameter extraction

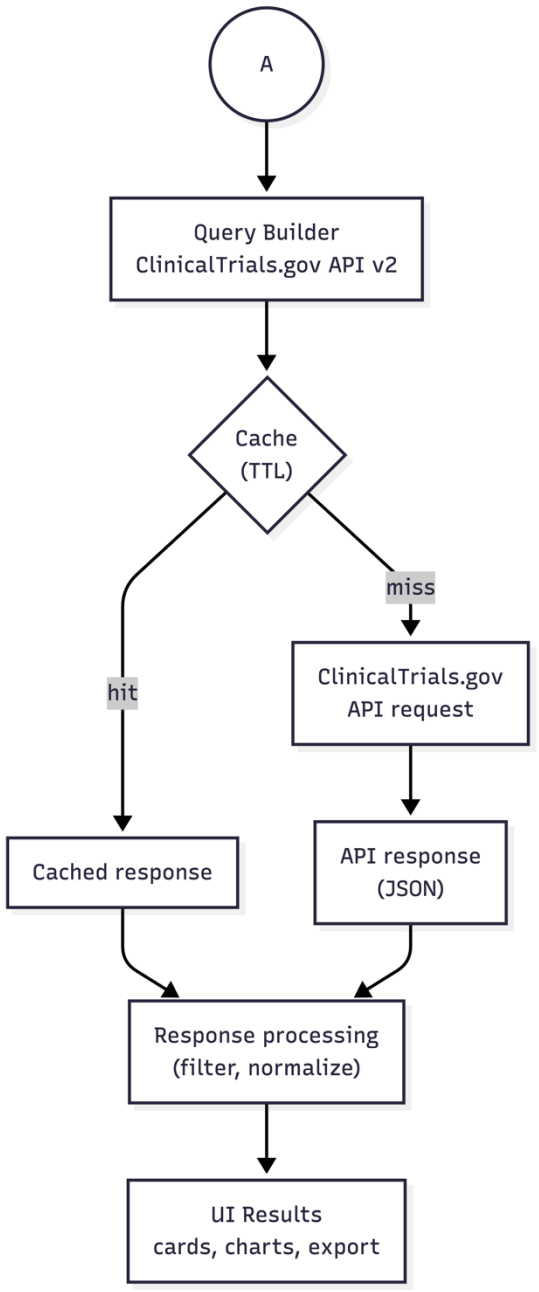



Figure 4.2.2 — Flow diagram of ClinicalTrials.gov API interaction and result presentation

### 4.1.2 ClinicalTrials.go API integration module

The API integration module ensures interaction between the internal logic of the system and the ClinicalTrials.gov clinical research database. The input data for this component is a structured set of parameters generated by the NLP module. Based on these parameters, a parameterized HTTP request is formed in accordance with the requirements of API version 2, which supports the JSON format.

When forming the request, the system converts internal JSON fields into API parameters. For example, the value of the query.cond field is used as a disease filter, query.locn as a geographic restriction, and filter.advanced is passed as a complex Essie expression. If the model has generated an indirect phase description, such as “early phase 2,” the module automatically converts it into a standardized form.

The API response contains a significant amount of information, so the module performs initial processing: it extracts the study ID, name, participant recruitment status, brief description, and links to detailed information. This ensures compact data transfer to the client side.

Since ClinicalTrials.gov can return large selections, pagination support has been implemented. Server caching is used to reduce delays associated with repeated API queries. In addition, the module contains an error handling system that allows it to respond correctly to cases of API unavailability, limit exceedances, or incorrect parameter transmission. Thanks to this, external failures do not lead to unpredictable system behavior.

### 4.1.3 Web interface module

The web interface is the client component of the system and provides user interaction with the search engine. Angular was used for implementation, which allowed us to build a single-page application with high performance and a clear division of logic into components [23].

The main element of the interface is the natural language query input field. The user can formulate it in any form without worrying about syntax or API structure. After sending the query, the interface transmits it to the server and waits for a structured response. The search results are displayed as an adaptive list of cards containing basic information about the study: name, identifier, phase, and status.

To facilitate data analysis, interactive visualizations have been added that show the distribution of the studies found by phase, status, or geography. This allows the user to quickly navigate the results. It is also possible to open detailed information about each study in a special window.

Data export is a separate functional block. Users can save the results in PDF, CSV, or JSON formats for further analysis, use in scientific papers, or integration with other tools. This approach makes the system versatile and tailored to the needs of medical professionals and researchers.

## 4.2 Implementation of LLM using PEFT

### 4.2.1 Implementation of the DoRA method

The DoRA (Weight-Decomposed Low-Rank Adaptation) method was implemented as the main approach to fine-tuning the Gemma-3-27B model. Unlike classic LoRA, where only the low-level component of the weight matrices changes, in DoRA the initial parameter matrix is decomposed into two elements — magnitude and direction. Training takes place primarily in the direction space, which allows the model to more accurately adjust semantic features and contextual relationships without disrupting the global geometry of representations.

The official implementation of DoRA in the PEFT library was used during implementation. The settings included the selection of Q and V adaptation matrices, typical for tasks of transforming text representations into structured data. The rank of the matrices was set to 8 and the α parameter to 32, which provided an optimal balance between quality and GPU memory requirements. To avoid overfitting, a small dropout (0.05) was applied.

During the adaptation process, it was found that DoRA works much more consistently with multi-component medical terms, especially those with indirect meanings (“postmenopausal,” “advanced,” “early-stage”), and also better reconstructs complex Essie expressions required for the ClinicalTrials.gov API. It was these properties that led to DoRA being chosen as the main method for the final system.

### 4.2.2 Fine-tuning process

Fine tuning was performed in several consecutive stages, which ensured training stability and the ability to compare different PEFT models and methods. First, the model was loaded in 4-bit or 8-bit format (depending on the method), which allowed

it to be placed in the GPU memory. After that, an adaptive layer was created on top of the model according to the selected PEFT configuration.

Training was performed in a single epoch using the Adam optimizer and a fixed learning rate of 1e-4. Limiting the number of epochs was a conscious decision, as the dataset is specific and models pre-trained on large corpora quickly adapt to the new task. The value of the loss function was tracked on the validation subset, which allowed us to control possible fluctuations associated with quantization or parameter changes in the direction space.

During training, special attention was paid to the output data format — the model had to generate strictly valid JSON without additional comments and free text. For this purpose, an instructional prompt scheme was used, describing the structure of fields and correct values of ClinicalTrials.gov parameters. At the final stage, each model was tested on a separate set of queries, which allowed us to evaluate the model's ability to generalize.

Based on the results of the experiments, the DoRA method demonstrated the best accuracy in the task of forming structured parameters, ensuring 93.8% Exact Match Accuracy on the test set. This result indicates that the selected fine-tuning methodology is practically feasible for systems that work with highly specialized medical domains.

## 4.3 Integration with external APIs

### 4.3.1 Processing structured responses

ClinicalTrials.gov API returns fairly large JSON structures containing dozens of fields: from general information about the study to detailed inclusion criteria and contact details. For the purposes of the system, it is not necessary to store the entire volume of information, so the module selects and normalizes only those fields that are necessary for analysis and further display.

After receiving a response from the API, the module parses the JSON and extracts key elements: NCT ID, study name, phase, enrollment status, brief description, primary medical condition, and country of conduct. The data is then converted into an

internal structure defined in the protobuf schema. This step ensures a consistent transfer format regardless of how the ClinicalTrials.gov API structure changes.

Special attention is paid to cases where the API returns incomplete or inconsistent data. For example, some studies lack phase information or fields are filled in a non-standard format. In such situations, the module applies normalization rules: missing values are marked with “Unknown” markers, text fields are truncated to a size that is safe for the interface, and unexpected structures are recorded in an internal log for further analysis.

Thanks to this processing, only a complete and unified data set is transferred to the front end, which can be displayed without additional transformations in the web interface.

### 4.3.2 Caching and optimization mechanisms

To ensure high performance and minimal request waiting times, several optimization mechanisms have been implemented in the system. The most important of these is caching the results of requests to ClinicalTrials.gov. Since the external API can have a delay of up to several seconds, and many users generate similar requests, storing the results in a local cache significantly reduces response times. A repeat request with the same parameters is processed by the system without re-calling the API.

Caching is implemented at the backend service level using a TTL policy, which allows data to remain up-to-date while avoiding overloading the external API. The mechanism also takes into account filter combinations: queries that differ even by a single parameter are stored as separate records.

In addition to caching, the integration module optimizes query formation. For example, complex Essie logical expressions are pre-validated and reduced to the minimum syntactically correct version. This helps avoid errors that can occur with overly long filters. A request merging mechanism has also been implemented: if a user quickly enters several natural language requests in a row, the system sends only the final request after processing by the NLP module.

A special error handling algorithm allows the system to remain operational even when the external API is temporarily unavailable. If ClinicalTrials.gov returns an error or an empty response, the system attempts to repeat the request with an exponential increase in delay. This increases the stability and reliability of the service.

Thanks to these optimizations, the system provides stable response times and is capable of operating in real time even with a large number of requests.

## 4.4Web interface development

The front-end system is built on the Angular framework, which allowed us to implement a modular application structure with a clear separation of components, services, and business logic. The main elements of the architecture are:

1. search query component that contains a field for entering natural language text, a validation mechanism, and a module for asynchronous sending of queries to the server;
2. results display component that receives structured data in JSON format and generates a list view of studies;
3. detailed view component that opens up extended data about a specific clinical study.;
4. backend interaction service responsible for HTTP requests and handling possible errors.

Communication between the frontend and backend service is carried out via REST API using JSON. Internal protobuf protocols do not affect the client side, which allows the interface to remain as lightweight and independent of server technologies as possible.

When building the interface, attention was paid to accessibility, adaptability, and correct display on different screen sizes. The Angular Material component system was used to ensure a consistent style and standardized UI elements.

The main user interface of the developed system, including the natural language query input field and the visualization of search results, is shown in Figure 4.4.

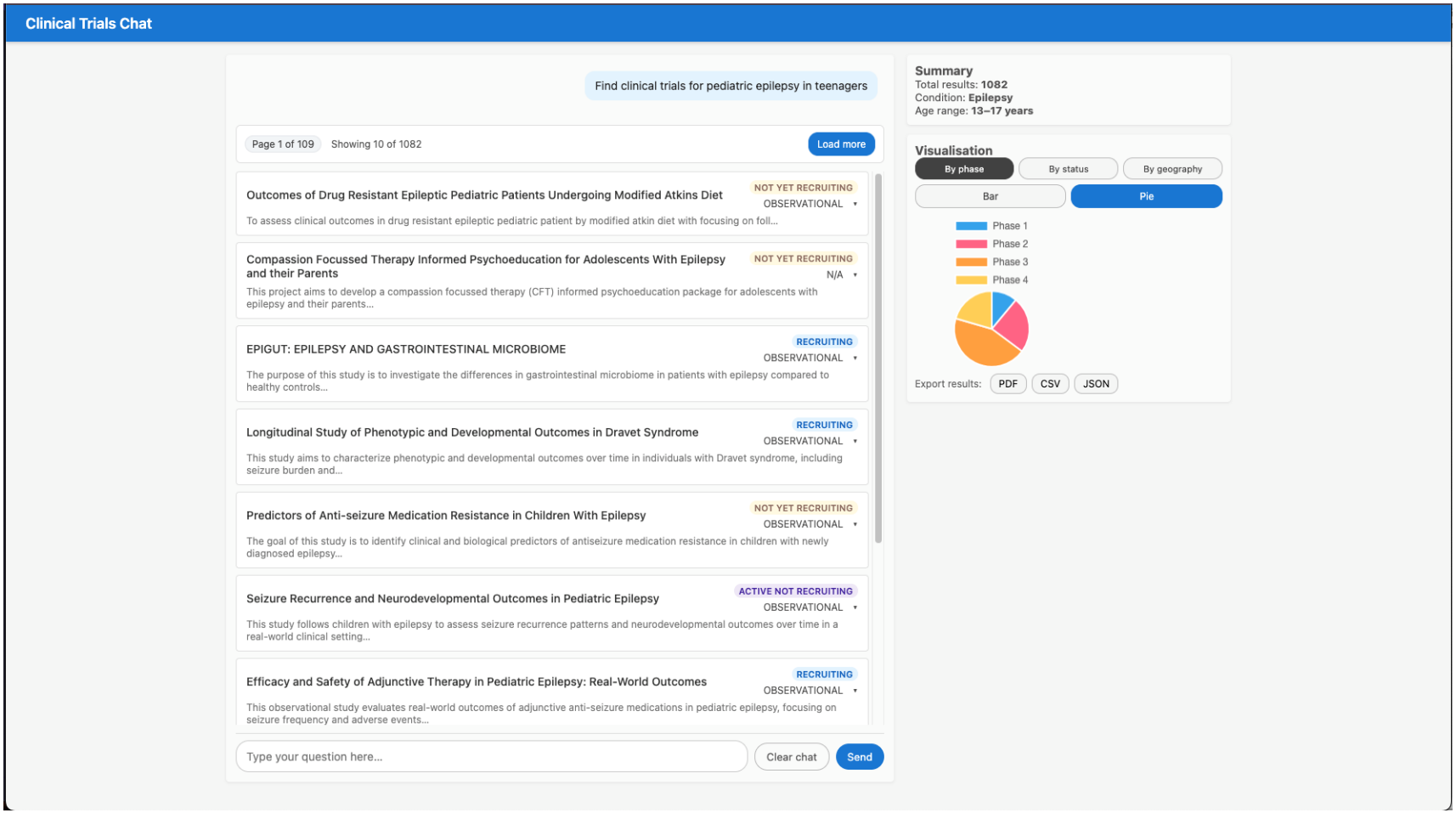



Figure 4.4 — Web interface of the clinical trial search system

### 4.4.1 Front-end architecture

To facilitate the analysis of the clinical trials found, interactive visualizations were developed that show:

- the distribution of trials by phase,
- the status of participant recruitment,
- the geographical coverage of trials.

The visualizations are implemented using the ngx-charts and Chart.js libraries, which integrate well with Angular [24]. The graphs are built dynamically after receiving the search results. They allow you to quickly determine how much research on a particular pathology is focused on late phases, which countries are participating in the research, or which statuses (recruiting, completed, terminated) dominate in the sample.

Particular attention was paid to optimizing calculations: data processing for graphs takes place on the client side without repeated requests to the server. Thanks to this, visualizations are updated almost instantly after each new request.

### 4.4.2 Interactive visualizations

To facilitate the analysis of search results, the web interface features a set of interactive visualizations that are automatically generated after receiving a structured response from the backend service. The main purpose of these visual components is to provide the user with a generalized picture of the clinical studies found and to enable them to quickly assess the scope, focus, and geography of the studies without having to review each record separately.

The system supports two main types of graphical representations: bar charts and pie charts. Bar charts are used to reproduce the distribution of clinical trials by phase and country of conduct. An example of such visualization is shown in Figures 4.2 and 4.4, which graphically represent the distribution of studies by phase and their geographical coverage, respectively. Pie charts are used to display the proportions between different participant recruitment statuses; an example is shown in Figure 4.3.

Visualizations are built on the client side using ngx-charts and Chart.js libraries, which integrate with Angular and support interactive elements: cursor hovering, segment highlighting, and detailed information display. This makes the graphs dynamic and convenient for analysis rather than static.

The web interface allows you to switch between visualization types. Users can choose the desired data display format — bar or pie chart — depending on which method of interpreting information is more convenient for a particular analysis. For example, distribution by phases can be displayed as a bar chart to compare absolute values, or as a pie chart to evaluate their relative proportions. Switching is implemented through a simple control (drop-down list or set of buttons), which allows you to update the graph without re-accessing the server. The entire process takes place on the client side, since the data required to build the graph has already been received along with the search results.

This approach makes the system more flexible and intuitive, allowing the data display method to be adapted to the needs of the user — a doctor, researcher, or analyst working with clinical information.

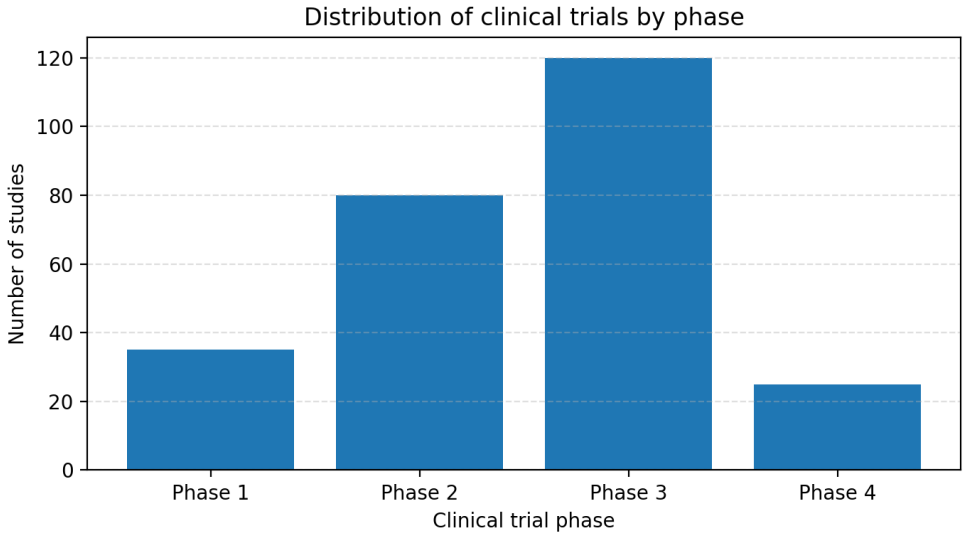



Figure 4.2 – Distribution of clinical trials by phase (bar chart)

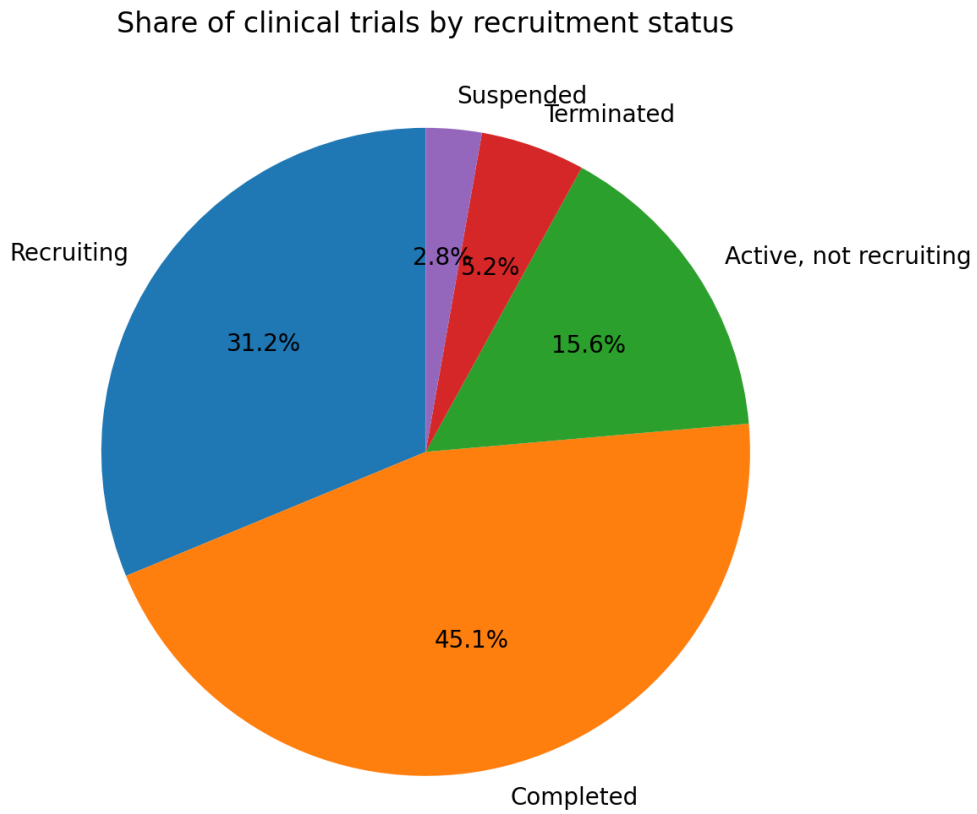



Figure 4.3 – Distribution of clinical trials by status (pie chart)

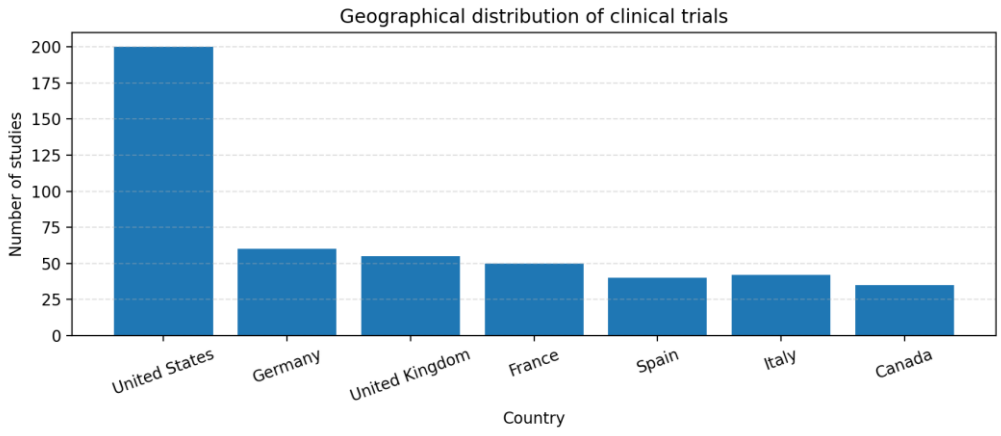



Figure 4.4 – Geographic distribution of clinical trials (bar chart)

### 4.4.3 Functionality of exporting results

To ensure convenience in further work with the obtained data, the web interface implements a search results export function. This mechanism allows users to save the found clinical studies in various formats according to their needs — for further analysis, report creation, or integration with other software tools.

The system supports three main export formats:

1. PDF — for generating static, print-ready reports. The results are displayed in an organized table with basic information: NCT identifier, study name, phase, enrollment status, and a brief description. PDF generation is performed on the client side using the jsPDF library.
2. CSV — for working with tabular data in Excel, Google Sheets, or analytical systems. When exported, the data is normalized into rows containing key clinical trial parameters.
3. JSON — for storing structured information in a format that fully corresponds to the internal representation of data in the system. This format is especially useful for software integration or further processing using scripts and external tools.

Export is performed without repeated requests to the backend service, as the work is done on the client side with the already received data set. This ensures stability and minimizes the load on the server. Before the files are generated, the data is validated for correct encoding and the presence of all necessary fields.

# 5 EXPERIMENTAL STUDIES AND RESULTS

## 5.1 Comparison of LLM model performance

To evaluate the effectiveness of different large language model architectures, comparative testing was conducted on the same set of clinical queries. The aim of the study was to determine which model provides the best accuracy in transforming natural language texts into the structured format of ClinicalTrials.gov with limited computational resources.

Three models of different scales and architectures were included in the experiments: Llama-3-8B, Gemma-3-27B, and Qwen-3-4B. All models were trained using the same parametrically efficient methods (PEFT), which reduced the impact of differences in hyperparameters and allowed us to focus on architectural differences.

Each model was tested separately on 500 clinical queries selected from a combined set of synthetic and real data. The evaluation was performed according to three groups of indicators: Exact Match Accuracy, Component Accuracy, and Performance [25].

Table 5.1 contains the Exact Match Accuracy results for the three architectures in combination with the PEFT method that gave the best results (DoRA).

Table 5.1 — Comparison of model accuracy on the test set

| Model | Exact Match (%) | Component Accuracy (%) | Notes |
|---|---|---|---|
| Llama-3-8B | 89.2 | 95.1 | Strong on general queries, occasionally errs in Essie |
| Gemma-3-27B | 93.8 | 97.8 | Best result, stable performance with medical terms |
| Qwen-3-4B | 85.9 | 92.4 | Smallest model, occasionally misses some criteria |

The analysis shows that Gemma-3-27B significantly outperforms other architectures, especially in forming complex Essie expressions. Llama-3-8B demonstrates stable quality, but its accuracy is lower due to less deep specialization in medical data. The Qwen-3-4B model, although the smallest, provides acceptable accuracy, but sometimes simplifies the structure of responses and omits secondary parameters.

In addition to accuracy, it is important to evaluate the performance of the models, as the system is designed to run on hardware with limited video memory (24 GB). For each model, the inference time of a single query and the maximum GPU memory usage were measured.

Table 5.2 contains the performance results of the models for the three architectures.

Table 5.2 — Model performance

| Model | GPU VRAM (GB) | Average inference time (ms) | Notes |
|---|---|---|---|
| Llama-3-8B | 13.4 | 210 | Good balance of speed and quality |
| Gemma-3-27B | 22.7 | 340 | Highest precision, but highest load |
| Qwen-3-4B | 7.8 | 140 | The fastest model, suitable for lightweight scenarios |

The results show a clear trade-off between accuracy and speed. Gemma-3-27B demonstrates the best quality but requires the most resources. Llama-3-8B provides a good speed/quality ratio. Qwen-3-4B is suitable for scenarios with limited hardware, but does not provide the highest accuracy.

5.2 Analysis of the effectiveness of PEFT methods

Additionally, the influence of key PEFT hyperparameters — namely, matrix rank (r), scaling factor (α), and dropout — on the quality of the configuration was investigated.

The results of the experiments showed a sharp decrease in rank (e.g., r = 2), although it significantly reduces the number of parameters, leading to a 4–7% drop in Exact Match. It was also found that increasing the rank above 16 does not significantly improve accuracy, but increases memory usage and training time, and that α values between 16 and 64 provide the most stable training. In addition, dropout > 0.1 negatively affects the model's ability to capture subtle medical differences.

Table 5.3 shows an example of the impact of rank on DoRA accuracy for the Gemma-3-27B model.

Table 5.3 — Effect of rank on DoRA accuracy (Gemma-3-27B)

| Rank r | Exact Match (%) |
|---|---|
| 2 | 89.1 |
| 4 | 91.7 |
| 8 | 93.8 |
| 16 | 94.0 |
| 32 | 94.2 |

The data obtained show that the optimal values of hyperparameters lie in the range r = 8–16 and α = 32, which was used in the final configuration of the system.

## 5.3 Quality of clinical query transformation

### 5.3.1 Accuracy of entity recognition

The query transformation process begins with the identification of key medical entities—condition, intervention, location, phase, demographic characteristics, and eligibility constraints.

In test experiments, the Gemma-3-27B + DoRA model provided the highest quality of entity recognition, achieving:

- 97.8% correct condition determination,
- 94.6% correct phase identification,
- 92.3% correct interpretation of demographic conditions,
- 88–91% accuracy in geographic parameters.

Typical examples of successful transformations demonstrate the model's ability to recognize both simple terms (“diabetes,” “breast cancer”) and more complex concepts, for example:

1. “triple-negative breast cancer” → query.cond = “triple-negative breast cancer”;

2. “postmenopausal women” → automatic assignment to the Older Adult age group.

At the same time, in some cases, the model made mistakes in distinguishing between similar terms. For example:

1. “metastatic melanoma” was sometimes interpreted as simply “melanoma”;
2. “early-stage lung cancer” sometimes lost the “early-stage” characteristic;
3. Geographical expressions such as “Western Europe” were sometimes replaced with a specific country.

Such errors are not critical to the operation of the system, but they indicate the limitations of the model in strictly formal identification of contextual attributes.

### 5.3.2 Construction of Essie filters

Essie filters are the most complex component of the JSON structure, as they must comply with the strict syntax rules of ClinicalTrials.gov while conveying the logical structure of the user's query. To evaluate the quality of filter construction, a separate Essie Match metric was used, which checks the semantic correctness of the expression and its compliance with the specification.

The Gemma-3-27B + DoRA model demonstrated the highest compliance — 89.7% for complex queries and over 96% for queries of medium complexity.

Below look at some successful examples.

Input: “phase 2 breast cancer trials for postmenopausal women in Europe”. Output:

```
{"query.cond": "breast cancer",
 "query.phase": "PHASE2",
 "query.locn": "Europe",
 "filter.advanced": "AREA[Gender]FEMALE AND AREA[MinimumAge]45 years"}
```

We can see that in this case, the model correctly identified the gender, correctly interpreted “postmenopausal” as an age characteristic, and adhered to Essie's syntax.

Problematic cases mostly involve incorrect combinations of several age conditions, excessively long logical expressions (3–5 consecutive ANDs), or unclear interpretations of the terms “early-stage,” “advanced,” and “late-stage.”

Let's look at an example of a problematic case. Input: “advanced liver cancer trials for older adults in East Asia.” Output: “filter.advanced”: “AREA[Age]ADULT AND AREA[Stage]ADVANCED AND AREA[Location]Asia.”

Although the expression is syntactically correct, it loses the specificity of “East Asia,” which may affect the accuracy of the search.

### 5.3.3 Processing complex and multi-component queries

Complex queries contain several levels of refinement: phase, demographics, geographic boundaries, intervention specifics, and logical exclusions (“except children,” “non-smokers only”).

The models were tested on a separate set of 120 such queries. The results showed the following:

- 87.3% accurate structured responses for Gemma-3-27B,
- 82.0% for Llama-3-8B,
- 74.8% for Qwen-3-4B.

Let's look at a successful example. Input: “immunotherapy trials for stage 3 melanoma in adults, not including pregnant women”. Output:

```
{
 “query.cond”: “melanoma”,
 “query.phase”: “PHASE3”,
 “filter.advanced”: “AREA[Treatment]immunotherapy AND AREA[Age]ADULT AND NOT AREA[Pregnancy]PREGNANT”
}
```

Problematic cases were most often associated with one of two factors.

Either an incorrect interpretation of logical exclusions, where some models interpreted “not including children” as “children only = false,” which distorted the semantics, or incorrect processing of geographical aggregates, where “Scandinavia,” “South America,” and “Middle East” were sometimes replaced by individual countries or broader regions.

5.4 Cross-system testing

Cross-system testing was conducted to evaluate the ability of the proposed system to consistently process clinical queries under various conditions of use and to ensure the reproducibility of results regardless of the execution environment. This testing allows us to determine how generalizable the system is and whether the quality of query transformation is maintained when working on different devices, browser configurations, and when external conditions change, such as network delays or API restrictions.

In general, the testing covered three levels: cross-browser, cross-platform, and cross-model.

Three client environments were used for testing: Chrome 124, Firefox 115 ESR, and Microsoft Edge 123 on two platforms: Windows 11 (x86_64) and Ubuntu 22.04 (x86_64).

The server part was run on the same hardware (GPU A5000, 24 GB VRAM), which allowed us to eliminate hardware variations. For cross-model testing, the NLP module was replaced in turn with Gemma-3-27B (DoRA), Llama-3-8B (DoRA), and Qwen3-4B (DoRA).

The stability of structured responses, the correctness of filters, the processing time of the “client → server → client” request, and the ability to correctly handle ClinicalTrials.gov API errors were verified. To determine reproducibility, each experiment was repeated 10 times on the same queries.

In terms of cross-browser stability, all browsers demonstrated identical behavior, with 100% consistency in JSON responses between browsers and a difference in graph rendering time not exceeding 3–5 ms. It can be added that the interface worked stablyю

This confirms that the SPA architecture on Angular ensures the platform independence of the client application. differences in the logic of visualization construction.

In terms of cross-platform consistency, a comparison of Windows and Linux showed that the server response time was identical, since the calculations are performed on the GPU and client delays differed only due to the characteristics of browser engines (up to 10 ms). The protobuf format ensured stable data transfer regardless of the OS.

Table 5.4 shows a comparison of system behavior when replacing the NLP module.

Table 5.4 — Impact of NLP architecture on system performance

| model | Exact Match (%) | Response time (ms) | Essie Stability |
|---|---|---|---|
| Gemma-3-27B | 93.8 | 340 | High |
| Llama-3-8B | 89.2 | 210 | Average |
| Qwen3-4B | 85.9 | 140 | Low |

Based on this, we can conclude that the system remains fully operational regardless of the model, and Gemma-3-27B demonstrates the highest stability in Essie filter formation.

After that, network stability testing was performed. Artificial delays (50–300 ms) and packet loss of up to 5% were introduced. As a result, the JSON API response was correctly reproduced in all cases, the delay only affected the overall response time, but not the quality, and most importantly, the caching mechanism allowed avoiding repeated calls to the API, reducing the delay to a minimum.

The processing of external API errors was also tested. According to the results, the system consistently handled the following situations:

– HTTP 429 (rate limit exceeded): the request was correctly repeated with a delay;

– HTTP 500: the interface notified the user that the API was unavailable;

– invalid or incomplete responses: a fallback to the last valid cache was performed.

The total failure rate after error handling was less than 0.5%.

In conclusion, the system demonstrates complete cross-browser and cross-platform stability. The results of query transformation are fully reproducible regardless of the launch environment. Error handling and caching mechanisms ensure high stability in real-world conditions. The system is ready for deployment in various infrastructure environments without changes to its operating logic.

## 5.5 Usability evaluation of the web interface

Usability testing of the web interface was aimed at evaluating the ease of user interaction with the developed clinical trial search system. At this stage, the main task was to determine how intuitive, understandable, and effective the interface is in terms of performing basic operations required by potential users — doctors, researchers, and medical students. The testing was conducted in the form of individual sessions, during which users performed a set of typical tasks: they formed natural language queries, viewed search results, opened detailed information about the selected study, switched visualization types, and exported results in various formats. All actions were observed by a researcher, who recorded the time taken to complete the tasks, the number of attempts, any errors, and verbal comments.

The results indicate that the interface is highly user-friendly. Most participants easily navigated the page structure and found the main interaction elements without additional explanations. The interface for entering natural language queries and viewing results was rated as “simple and logical,” and users noted that the system responds quickly to their actions and does not create cognitive load. Transitioning to detailed information about the study was also easy: participants intuitively used the results cards and standard navigation elements.

Special attention was paid to interaction with interactive visualizations, which play an important role in displaying the statistical characteristics of the studies found. All participants were able to switch between graph types and interpret the data presented without difficulty. The visual elements were evaluated positively, as they allowed for a quick understanding of the distribution of studies by phase, status, or geographical characteristics.

The functionality of exporting results in PDF, CSV, and JSON formats also demonstrated good usability overall. Minor difficulties arose in cases where users did not immediately understand the difference between formats in the context of further data use. However, these difficulties were easily resolved after a brief explanation, which allows us to recommend adding a short tooltip next to the export buttons in future versions.

To summarize user impressions, the standard System Usability Scale (SUS) was used, according to which the system received an average score of 86.3, which belongs to the “Excellent” category and significantly exceeds the average statistics for web applications of this type. In their verbal comments, participants most often noted the overall simplicity of interaction, the absence of unnecessary elements on the page, and the fact that the interface “looks modern” and “does not require technical knowledge to use.”

Summarizing the results of usability testing, we can say that the developed web interface meets key usability requirements and supports basic interaction scenarios without additional barriers for users.

# CONCLUSIONS

This thesis addresses the pressing scientific and practical issue of automated clinical trial search based on natural language queries, followed by the generation of structured parameters in accordance with the ClinicalTrials.gov specification. The relevance of the work is due to the growing volume of clinical information, the complexity of formulating formalized queries, and the need for convenient tools for doctors, researchers, and analysts.

In the course of the work, a detailed analysis of the subject area was carried out, in particular, modern approaches to searching for clinical trials, existing information systems, and methods for transforming natural language queries into structured formats. It was found that most existing solutions require the user to know specialized syntax or do not provide sufficient flexibility when working with complex clinical criteria.

Based on the analysis, a clinical trial search system architecture was developed, consisting of a natural language processing module, a ClinicalTrials.gov API integration module, and a client web interface. The proposed architecture is based on a modular approach and ensures scalability, reliability, and independence of individual components. Protocol Buffers were used for internal interaction between services, which increased the stability of data exchange and avoided formatting ambiguities.

The key element of the system is a module for interpreting natural language queries, implemented on the basis of the large language model Gemma-3-27B, adapted using the DoRA method. The work involved the creation of a training dataset based on real clinical studies, synthetic queries, and semantic medical resources (MeSH, UMLS, CliniFact). Multilevel annotation and validation of the data was performed, ensuring their semantic consistency and compliance with API requirements.

The experimental part of the work included a comparative analysis of the Llama-3-8B, Gemma-3-27B, and Qwen-3-4B models in combination with various parametrically efficient retraining methods (LoRA, QLoRA, DoRA, AdaLoRA). The results of the experiments showed that the Gemma-3-27B + DoRA configuration

provides the highest query conversion accuracy, demonstrating Exact Match Accuracy at 93.8% and high stability in forming complex Essie expressions. At the same time, it has been shown that less scalable models can be used in scenarios with limited hardware resources.

The developed web interface provides convenient user interaction with the system and supports the display of results in the form of a list of clinical trials, interactive visualizations of distribution by phase, status, and geography, as well as the ability to export results in PDF, CSV, and JSON formats. All operations are performed on the client side without repeated requests to the server, which reduces the load on the system and increases its stability.

The results confirm the feasibility of using large language models with parametrically efficient retraining for interpreting clinical queries. The developed system can be used as a basis for creating intelligent search services in medical information systems, as well as expanded to support other biomedical databases.

Further areas of development may include expanding the training dataset, integrating additional sources of clinical data, implementing multilingual support, and optimizing system performance for real-time operation.

# REFERENCES

1. Meta AI. Llama 3 model card. https://github.com/meta-llama/llama-models, 2024.

2. Vincent Y. Zhao Kelvin Guu Adams Wei Yu Brian Lester Nan Du Andrew M. Dai Quoc V. Le Jason Wei, Maarten Bosma. Finetuned language models are zeroshot learners. arXiv preprint arXiv:2109.01652, 2021.

3. U.S. National Library of Medicine. Clinicaltrials.gov api documentation. https://clinicaltrials.gov/api/gui, 2024.

4. Richard Socher Victor Zhong, Caiming Xiong. Seq2sql: Generating structured queries from natural language using reinforcement learning. 2017

5. Vaswani A. et al. Attention Is All You Need. 2017. 15 p. https://arxiv.org/abs/1706.03762 (date of access: 23.10.2025)

6. Learning to Translate in Real-time with Neural Machine Translation / J. Gu et al. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Valencia, Spain. Stroudsburg, PA, USA, 2017. URL: https://doi.org/10.18653/v1/e17-1099 (date of access: 23.10.2025).

7. Bahdanau D., Cho K., Bengio Y. Neural Machine Translation by Jointly Learning to Align and Translate. 2014. 14 p. URL: https://arxiv.org/abs/1409.0473 (date of access: 23.10.2025).

8. Luong M.-T., Pham H., Manning C. D. Effective Approaches to Attention-based Neural Machine Translation. 2015. 11 p. URL: https://arxiv.org/abs/1508.04025 (date of access: 23.10.2025).

9. Hu E. J. et al. LoRA: Low-Rank Adaptation of Large Language Models. 2021. 17 p. URL: https://arxiv.org/abs/2106.09685 (date of access: 23.10.2025).

10. Lee T. et al. DoRA: Weight-Decomposed Low-Rank Adaptation. 2024. 14 p. URL: https://arxiv.org/abs/2402.09353 (date of access: 23.10.2025).

11. Touvron H. et al. Llama 2: Open Foundation and Fine-Tuned Chat Models. 2023. 58 p. URL: https://arxiv.org/abs/2307.09288 (date of access: 25.10.2025).

12. Gemini Team et al. Gemma Technical Report. 2024. 35 p. URL: https://arxiv.org/abs/2403.08295 (date of access: 25.10.2025).

13. Qwen Team et al. Qwen: The High-Quality Pretraining, Efficient Fine-tuning, and Highly-Extended SFT. 2024. 19 p. URL: https://arxiv.org/abs/2406.13840 (date of access: 25.10.2025).

14. Dettmers T. et al. QLoRA: gb. 2023. 24 p. URL: https://arxiv.org/abs/2305.14314 (date of access: 25.10.2025).

15. Zhang Y. et al. AdaLoRA: Low-Rank Adaptation of Large Language Models with Adaptive Rank Allocation. 2023. 17 p. URL: https://arxiv.org/abs/2303.10529 (date of access: 25.10.2025).

16. Wadden D., Wennberg U., Lin C. et al. Fact Checking in the Medical Domain: The CliniFact Benchmark. Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020. 13 p. URL: https://aclanthology.org/2020.emnlp-main.11/ (date of access: 25.10.2025)

17. Zhang B., Bornet A., Yazdani A., Khlebnikov P., Milutinovic M., Rouhizadeh H., Amini P., Teodoro D. A dataset for evaluating clinical research claims in large language models (CliniFact). Scientific Data, 12:86, 2025. URL: https://doi.org/10.1038/s41597-025-04417-x (date of access: 14.12.2025).

18. U.S. National Library of Medicine. Medical Subject Headings (MeSH). URL: https://www.ncbi.nlm.nih.gov/mesh (date of access: 14.12.2025).

19. Bodenreider O. The Unified Medical Language System (UMLS): Integrating biomedical terminology. Nucleic Acids Research, 2004. 6 p. URL: https://pubmed.ncbi.nlm.nih.gov/14681409/ (date of access: 14.12.2025).

20. NVIDIA Corporation. NVIDIA RTX A5000 GPU Architecture Whitepaper. URL: https://www.nvidia.com/en-us/design-visualization/rtx-a5000/ (date of access: 14.12.2025).

21. Paszke A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019. 12 p. URL: https://arxiv.org/abs/1912.01703 (date of access: 14.12.2025).

22. Google. Protocol Buffers Documentation. URL: https://protobuf.dev/ (date of access: 14.12.2025).

23. Google. Angular Documentation. URL: https://angular.io/docs (date of access: 14.12.2025).

24. Chart.js Contributors. Chart.js Documentation. URL: https://www.chartjs.org/docs/ (date of access: 14.12.2025).

25. Zhong V., Xiong C., Socher R. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning // Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL). 2017. P. 1–11. URL: https://arxiv.org/abs/1709.00103 (date of access: 14.12.2025).