

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
ПЕТРА ВЕЛИКОГО

Физико-механический институт

Высшая школа прикладной математики и вычислительной физики

Отчёт по лабораторной работе №7

по дисциплине

"Автоматизация научных исследований"

**Поиск научной литературы**

Выполнил студент:

Лапина Ольга Константиновна

Группа 5040102/50201

Санкт-Петербург  
2025

## Содержание

1. Задание.....	3
2. Промпты.....	4
2.1. Промпт 1.....	4
2.2. Промпт 2.....	4
2.3. Промпт 3.....	5
2.4. Промпт 4.....	5
3. Результаты.....	6
3.1. Использование предложенных промптов.....	6
3.1.1. Промпт 1.....	6
3.1.2. Промпт 2.....	7
3.1.3. Промпт 3.....	10
3.1.4. Промпт 4.....	12
4. Сравнительный анализ результатов.....	15
Таблица 1: Оценка ответов ИИ-моделей.....	15
4.1. Анализ сильных и слабых сторон моделей.....	19
Промпт 1: Простые поисковые запросы.....	19
Промпт 2: Пакеты запросов (Basic + Soft-phrase).....	20
Промпт 3: От общего к точному.....	21
Промпт 4: Фасетный поиск на основе текста.....	21
5. Разработка собственного промпта (Оптимизация).....	22
5.1. Применение промпта.....	25
5.2. Оценка финального ответа.....	25
6. Выводы.....	27

## **1. Задание**

В рамках лабораторной работы необходимо произвести:

1. Подготовка исходной статьи (желательно по теме НИР/ВКР).
2. Выбор моделей. Выбрать две различные ИИ-модели для проведения эксперимента.
3. Эксперимент. Отправить текст статьи моделям,используя каждый из четырех предложенных промптов.
4. Анализ результатов. Провести детальный сравнительный анализ 8 полученных ответов, выявляя сильные и слабые стороны различных моделей при работе с различными промтами.
5. Оптимизация. Выбрать одну ИИ-модель, которая, по вашему мнению, показала наилучший потенциал, и на основе выявленных недостатков существующих промптов, составить собственный промпт, который, по вашему мнению, должен дать наилучший результат по всем критериям.
6. Финальная проверка. Прогнать статью через выбранную модель, используя свой авторский промпт.

### **Критерии оценивания ответов ИИ**

- Достоверность и проверяемость источников
- Релевантность теме
- Покрытие подтем (полнота)
- Разнообразие без дублей
- Воспроизводимость и прозрачность (можно ли повторить ваш результат и проверить путь получения)

## 2. Промпты

*\*Промпты поправлены в соответствии с выбранной темой (обозначена в п.3)*

### 2.1. Промпт 1.

Сгенерируй 12 простых поисковых запросов для сайта arXiv (<https://arxiv.org/search>), которые точно дают выдачу.

Требования:

- Английский язык, 2–4 общих ключевых слова без кавычек, без двоеточий и логических операторов.
- Не используй редкие аббревиатуры и длинные составные фразы.
- В каждом запросе должен быть хотя бы один «якорный» термин: interface, API, software, design, Python, binding, extension, system, model, method, analysis, dataset — выбери подходящие к теме.
- Выведи только готовые строки запросов, по одному на строку.

### 2.2. Промпт 2.

Подготовь 10 пакетов запросов для сайта arXiv (<https://arxiv.org/search>). Каждый пакет — два web-safe варианта, которые дают выдачу:

- Basic: 3–6 ключевых слов без кавычек и операторов (только пробелы).
- Soft-phrase: одна короткая фраза в кавычках до 3 слов + 1–2 общих слова (например: "graph neural" models).

Ограничения:

- Только английский; избегай редких узких терминов и длинных фраз.
- Не используй NOT/AND/OR, поля (ti:, abs:, cat:) и двоеточия.
- В каждом варианте добавляй 1 «якорное» слово (interface, API, software, design, Python, binding, extension, system, model, method, analysis, dataset).

Формат вывода: для каждого пакета две строки подряд — сначала Basic, затем Soft-phrase.

### 2.3. Промпт 3.

Сформируй 10 наборов эскалируемых запросов для arXiv по теме: {тема}.  
Для каждого набора дай три строки:

- Q0 (web-safe): 3–5 общих ключевых слов без кавычек и операторов — ориентирован на <https://arxiv.org/search> (All fields).
- Q1 (web-safe+): одна короткая фраза в кавычках (до 3 слов) + 1–2 общих слова. Без операторов.
- Q2 (fielded мягкий, для API или Advanced Search): abs:"<краткая фраза до 3 слов>" OR ti:"<краткая фраза до 3 слов>" — без NOT; категорию не добавляй.

Правила:

- Только английский; избегай редких аббревиатур (кроме общеизвестных вроде API, NLP).
- Не используй длинные точные фразы.
- Не показывай рассуждения; выведи только тройки строк в порядке Q0, Q1, Q2 для каждого набора.

### 2.4. Промпт 4.

Выдели 6 фасетов (подтем) по тексту ниже (каждый 2–3 слова), НО не выводи их отдельно. Для каждого фасета сгенерируй по 3 запроса, предназначенные для arXiv и дающие выдачу:

- Minimal (web-safe): 2–4 общих ключевых слова, без кавычек и операторов.
- Broadened (web-safe): добавь 1–2 синонима через пробел (без OR), оставь якорные слова (interface, API, software, design, Python, binding, extension, system, model, method, analysis, dataset).
- Review-oriented (web-safe): включи слова survey или review вместе с темой, без кавычек и операторов.

Ограничения:

- Только английский. Запрещены NOT/AND/OR, поля (ti:, abs:, cat:), двоеточия и длинные точные фразы.
- Избегай узких редких терминов; предпочитай общеупотребимые.

Формат вывода:

- 18 строк (по 3 на каждый из 6 фасетов) в группах по 3 строки подряд: Minimal, затем Broadened, затем Review-oriented.

{текст}

### 3. Результаты

В качестве **темы** выберу скорректированную тему своей бакалаврской работы: “Разработка программных интерфейсов в современных инженерных приложениях.”

В качестве описания использовался реферат и ключевые слова к ВКР:

Данная работа посвящена исследованию принципов разработки языковых программных интерфейсов для современных инженерных приложений с фокусом на программные комплексы для моделирования многофазных потоков. Основная цель заключается в проектировании и реализации программного интерфейса на Python с использованием Python/C API, обеспечивающего дублирование функциональности графического интерфейса пользователя. Проведен анализ современных подходов к проектированию API для инженерного ПО, включающий: классификацию API; сравнительный анализ технологий связывания языков; основные принципы проектирования языкового программного интерфейса; особенности проектирования на Python/C API. В результате были разработаны модули расширения, позволяющие работать с объектами разрабатываемого программного комплекса посредством скриптовых команд Python, а не только через графический интерфейс. Проведено тестирование полученных модулей с использованием рабочих скриптов, представленных в работе. С помощью разработанного программного интерфейса на Python можно выполнять циклические действия с моделью и расчётами, а также интегрировать сторонние библиотеки (NumPy, SciPy, Matplotlib) для расширения функциональности.

Ключевые слова: ПРОГРАММНЫЕ ИНТЕРФЕЙСЫ, ЯЗЫКОВЫЕ ПРОГРАММНЫЕ ИНТЕРФЕЙСЫ, PYTHON/C API, МОДЕЛИРОВАНИЕ МНОГОФАЗНОГО ПОТОКА, ПРОГРАММНЫЙ КОМПЛЕКС

Ссылка на ВКР: <https://elib.spbstu.ru/dl/3/2025/vr/vr25-2727.pdf/info>

**Модели:** DeepSeek и ChatGPT

#### 3.1. Использование предложенных промптов

##### 3.1.1. Промпт 1.

###### 1. DeepSeek

**Ответ:**

software design patterns Python  
computer vision model analysis  
API interface system  
quantum computing method  
dataset machine learning  
neural network model training  
natural language processing software  
database management system  
Python data analysis  
graph neural network extension  
optimization algorithm method  
image processing software

## 2. ChatGPT

**ОТВЕТ:**

software design Python  
Python API interface  
system model dataset  
method analysis system  
software API extension  
Python binding interface  
model analysis dataset  
design method software  
API system software  
Python method analysis  
dataset interface design  
software extension binding

### 3.1.2. Промпт 2.

#### 1. DeepSeek

**ОТВЕТ:**

1.  
machine learning Python framework  
"deep learning" Python framework

2.  
quantum computing software simulation  
"quantum algorithms" software simulation
3.  
data visualization Python library  
"interactive plots" Python library
4.  
computer vision model dataset  
"object detection" model dataset
5.  
natural language processing method analysis  
"text classification" method analysis
6.  
cloud computing system design  
"serverless computing" system design
7.  
optimization algorithms Python package  
"gradient descent" Python package
8.  
neural network API binding  
"transformer models" API binding
9.  
statistical analysis software tool  
"time series" analysis software
10.  
web application design interface  
"user interface" design system

## 2. ChatGPT

**Ответ:**

Вот 10 пакетов запросов для arXiv, каждый из которых содержит



**\*\*две строки\*\*** — сначала **\*\*Basic\*\***, затем **\*\*Soft-phrase\*\***. Все запросы на английском, без логических операторов и двоеточий, и в каждом есть одно «якорное» слово из указанного списка.

1.

basic: neural network design interface performance  
soft-phrase: "neural network" design interface

2.

basic: python data analysis method performance  
soft-phrase: "data analysis" python method

3.

basic: software testing dataset quality assurance  
soft-phrase: "testing dataset" software quality

4.

basic: graph model binding learning representation  
soft-phrase: "graph model" binding learning

5.

basic: api integration system reliability evaluation  
soft-phrase: "api integration" system reliability

6.

basic: software extension method usability study  
soft-phrase: "extension method" software usability

7.

basic: system performance model optimization techniques  
soft-phrase: "performance model" optimization techniques

8.

basic: dataset collection python tool benchmarking

soft-phrase: "dataset collection" python tool

9.

basic: interface design user experience research

soft-phrase: "user experience" interface design

10.

basic: method validation analysis results interpretation

soft-phrase: "validation analysis" method results

Если нужно адаптировать под конкретную тему (например, NLP, robotics, bioinformatics), могу пересобрать запросы более целевыми!

### 3.1.3. Промпт 3.

#### 1. DeepSeek

**Ответ:**

software engineering API design

"API design" software engineering

abs:"API design" OR ti:"API design"

REST API software applications

"REST API" engineering

abs:"REST API" OR ti:"REST API"

microservices architecture engineering

"microservices architecture" design

abs:"microservices architecture" OR ti:"microservices architecture"

software integration interfaces

"software integration" interfaces

abs:"software integration" OR ti:"software integration"

engineering simulation API

"simulation API" engineering

abs:"simulation API" OR ti:"simulation API"

computer aided design API  
"CAD API" development  
abs:"CAD API" OR ti:"CAD API"

industrial software interfaces  
"industrial software" interfaces  
abs:"industrial software" OR ti:"industrial software"

model driven engineering API  
"model driven" API  
abs:"model driven" OR ti:"model driven"

engineering web services  
"web services" engineering  
abs:"web services" OR ti:"web services"

computational engineering interface  
"computational interface" software  
abs:"computational interface" OR ti:"computational interface"

## 2. ChatGPT

### Ответ:

software interfaces engineering applications APIs  
"software interfaces" engineering applications  
abs:"software interfaces" OR ti:"software interfaces"

software architecture engineering systems APIs  
"engineering systems" software architecture  
abs:"engineering systems" OR ti:"engineering systems"

API design engineering software tools  
"API design" engineering tools  
abs:"API design" OR ti:"API design"

interface development engineering software  
"interface development" engineering  
abs:"interface development" OR ti:"interface development"

software integration engineering platforms

"software integration" engineering platforms  
abs:"software integration" OR ti:"software integration"

engineering simulation software interfaces  
"simulation software" interfaces  
abs:"simulation software" OR ti:"simulation software"

REST API engineering applications design  
"REST API" engineering design  
abs:"REST API" OR ti:"REST API"

user interfaces engineering applications  
"user interfaces" engineering  
abs:"user interfaces" OR ti:"user interfaces"

modular interfaces engineering frameworks  
"modular interfaces" engineering  
abs:"modular interfaces" OR ti:"modular interfaces"

software interoperability engineering systems  
"software interoperability" engineering  
abs:"software interoperability" OR ti:"software interoperability"

### 3.1.4. Промпт 4.

#### 1. DeepSeek

**ОТВЕТ:**

Python API design  
software interface Python binding  
survey API design Python

engineering software APIs  
engineering software API Python integration  
review engineering software APIs

multiphase flow simulation  
multiphase flow modeling software  
survey multiphase flow software

language binding techniques  
software extension language binding  
review language binding methods

Python C extension  
Python C API module  
review Python C extensions

scripting interface automation  
Python scripting simulation automation  
review simulation scripting interfaces

## 2. ChatGPT

**Ответ:**

Вот 18 запросов (6 фасетов  $\times$  3 типа) для **arXiv**, по вашему тексту. Они в нужном формате — только английские слова, без операторов, двоеточий и узких терминов.

---

### 1) Language API design

1. language API design
2. language interface API design system
3. language API design survey review

### 2) Python C API

4. Python C API
5. Python API C binding extension
6. Python C API review survey

### 3) Engineering software interfaces

7. engineering software interface
8. engineering software interface design system
9. engineering software interface survey review

### 4) Multiphase flow modeling

10. multiphase flow modeling
11. multiphase flow simulation model method
12. multiphase flow review survey

## **5) Scriptable modules Python**

- 13. Python scriptable modules extension
- 14. Python scriptable modules API software
- 15. Python scriptable modules review survey

## **6) Integrating scientific libraries**

- 16. scientific libraries integration Python
- 17. scientific libraries integration system API
- 18. scientific libraries review survey

---

Если понадобится адаптация под конкретные категории arXiv — например для cs.SE или physics.comp-ph — могу сделать!

Модель	Достоверно сть и проверяемо сть источников	Релевантность теме	Покрытие подтем (полнота)	Разнообразие без дублей	Воспроизводи мость и прозрачность	Итоговая оценка
Промпт 1						
DeepSeek	<b>Высокая.</b> Все источники — научные препринты с arXiv.org или его официальна я документация, имеют идентификаторы и даты.	<b>Низкая.</b> Только 1 запрос из 12 напрямую о разработке API. Остальные — о смежных областях (алгоритмы, ML, прикладное ПО).	<b>Низкая.</b> Освещена лишь одна подтема — практическая работа с конкретным веб-API (arXiv). Ключевые аспекты проектирования API не затронуты.	<b>Высокая.</b> Результаты представлены разными форматами: исследования, обзоры, документация. Явных дублей нет.	<b>Средняя.</b> Для документации API — полная. Для научных статей — зависит от описания методологии в каждой работе.	<b>Низкая.</b> Набор запросов плохо сфокусирован. Он дает общий контекст (паттерны, данные, алгоритмы), но не позволяет системно изучить разработку программных интерфейсов (API).
ChatGPT	<b>Высокая.</b> Все источники — научные препринты с arXiv.org, имеющие уникальные ID, авторов, даты и аннотации.	<b>Средняя.</b> 5 из 12 запросов прямо содержат ключевые слова "API" или "interface". Результаты более сфокусированы на проектировании, паттернах и системах, но часто в контексте ML, а не чистого API-дизайна.	<b>Средняя.</b> Затронуты узкие практические аспекты: создание биндингов (Python binding interface), расширение систем (software extension binding).	<b>Низкая.</b> В результатах высокая доля дублей и погранично похожих статей. Запросы семантически пересекаются (software design Python / design method software; system model dataset / model analysis dataset), что ведет к повторяющимся результатам.	<b>Средняя.</b> Для статей по API arXiv и некоторым паттернам — хорошая, есть примеры кода. Для исследовательских статей по ML — стандартная для научных препринтов (зависит от детальности метода).	<b>Средняя.</b> Ключевые слова "API", "interface", "design", "binding" прямо связаны с разработкой интерфейсов. Системного покрытия принципов проектирования API нет.

Промпт 2						
DeepSeek	<b>Высокая.</b> Все источники — научные препринты с arXiv.org. Каждый документ имеет уникальный идентификатор (arXiv ID), дату, авторов и постоянную ссылку, что обеспечивает полную проверяемость	<b>Средняя.</b> Запросы сфокусированы на инструментах и инфраструктуре ПО (фреймворки, библиотеки), что косвенно связано с разработкой интерфейсов. Однако прямой фокус на проектировании API, контрактах или архитектурных паттернах для интерфейсов встречается редко.	<b>Низкая</b> Освещены в основном узкие аспекты: создание специализированных фреймворков для экспериментов или предметных областей, инструменты для визуализации и анализа данных. Ключевые подтемы проектирования интерфейсов (протоколы, версионирование, документация) не затронуты.	<b>Средняя.</b> Результаты охватывают широкий спектр областей: машинное обучение (MLXP, PyTorch), квантовые вычисления, биоинформатика, компьютерное зрение. Дублирования информации между ними нет.	<b>Средняя.</b> Для статей, представляющих конкретные библиотеки (MLXP, DataSist, EasyVitesce), часто есть ссылки на публичный код, что облегчает проверку. Для концептуальных работ (о программном стеке для квантового моделирования) воспроизводимость зависит от реализации описанных идей.	<b>Средняя.</b> Представленный набор парных запросов эффективно фокусируется на поиске конкретных программных инструментов, фреймворков и библиотек, которые являются конечным продуктом разработки ПО. Это позволяет находить актуальные исследования о реализации программных систем.
ChatGPT	<b>Высокая.</b> Как и в предыдущих случаях, все источники — научные препринты с arXiv.org. Каждый документ имеет идентификатор, аннотацию, список авторов, что позволяет проверить информацию и найти исходную работу.	<b>Средняя.</b> В запросах прямо используются термины "interface", "api integration", "binding", что формально соответствует теме. Однако контекст их применения часто смещен: "interface" может означать интерфейс пользователя или абстрактную прослойку в системах ML, а не программный API. Запросы о дизайне и валидации методов также касаются темы лишь косвенно.	<b>Низкая.</b> Результаты раскрывают лишь очень узкие аспекты: создание биндингов для работы с графовыми моделями, дизайн интерфейсов для визуализации данных. Основные инженерные практики проектирования API (архитектурные стили, управление версиями, безопасность) не представлены.	<b>Средняя.</b> Набор запросов охватывает разные области (ML, анализ данных, системная интеграция, UX), что обеспечивает общее разнообразие результатов. Однако внутри некоторых пар (например, запросы 1 и 2) результаты сильно пересекаются, так как касаются смежных тем (производительность в ML и анализ данных).	<b>Средняя.</b> Для работ, связанных с конкретными инструментами и фреймворками (например, GraphST для анализа пространственной транскриптомики), воспроизводимость зависит от наличия кода и данных. Для теоретических работ о дизайне методов или интерфейсов — проверка сводится к анализу предложенных концепций.	<b>Средняя.</b> Результаты отражают не разработку программных интерфейсов как инженерную дисциплину, а их вторичную роль в контексте других исследований (машинного обучения, анализа данных, системной интеграции). Запросы фокусируются на применении или тестировании интерфейсов, а не на принципах их создания.



Промпт 3						
DeepSeek	<b>Высокая.</b> Результаты по всем запросам — научные препринты с arXiv (разделы cs.SE, cs.DC и др.). Каждая статья имеет идентификатор, аннотацию и ссылку, обеспечивая полную проверяемость.	<b>Высокая.</b> Запросы напрямую и точно нацелены на ключевые аспекты темы: проектирование API, архитектурные стили (REST, микросервисы), интеграцию, применение в инженерных областях (CAD, симуляции). Результаты прямо соответствуют формулировкам.	<b>Средняя.</b> Набор хорошо освещает архитектурные и предметно-ориентированные (CAD, симуляции, промышленное ПО) аспекты разработки интерфейсов. Однако не хватает запросов на такие ключевые подтемы, как управление жизненным циклом API (версионирование, депривация), безопасность, инструменты и стандарты (OpenAPI, gRPC), производительность и тестирование API.	<b>Высокая.</b> Разные запросы ведут к разным кластерам статей: от фундаментальных принципов проектирования до узкоспециализированных API для CAD или промышленных систем. Значимого дублирования между результатами по разным наборам не наблюдается.	<b>Высокая.</b> Использование операторов abs: и ti: для поиска в заголовках и аннотациях делает поиск на arXiv максимально точным и воспроизводимым. Любой пользователь, введя эти запросы в Advanced Search, получит идентичные результаты.	<b>Высокая.</b> Запросы: -Прямо соответствуют теме: используют точную терминологию ("API design", "REST API", "microservices architecture"). -Охватывают важные подразделы: от общих принципов до специализации в инженерных доменах. -Технически грамотны: эскалация от общего (Q0) к уточненному (Q1) и целевому поиску в метаданных (Q2) — это лучшая практика для академического поиска.
ChatGPT	<b>Высокая.</b> Результаты по всем запросам — научные препринты с arXiv (в основном cs.SE, cs.DC, cs.CE). Каждая статья имеет идентификатор (arXiv ID), аннотацию, ссылки и авторов, обеспечивая полную	<b>Средняя.</b> Основной набор запросов ("API design", "REST API", "software integration", "interface development") — высоко релевантен. Однако включение запросов о "user interfaces" (пользовательские интерфейсы, UI/UX) и "software architecture" (без явной привязки к интерфейсам) снижает общую	<b>Средняя.</b> Набор хорошо освещает ядро темы: проектирование API, интеграцию, REST. Запрос про "modular interfaces" затрагивает важный аспект модульности. Однако, как и в прошлый раз, не хватает запросов на специфические инженерные подтемы: управление жизненным	<b>Средняя.</b> Разные запросы ведут к разным кластерам статей, что обеспечивает разнообразие. Однако наблюдается семантическое пересечение и потенциальное дублирование результатов между некоторыми запросами (например, "software interfaces" и "interface	<b>Высокая.</b> Использование операторов abs: и ti: для поиска в заголовках и аннотациях делает поиск на arXiv точным и полностью воспроизводимым. Любой пользователь получит идентичные результаты.	<b>Средняя/Высокая.</b> н фокусируется на ключевых инженерных аспектах (разработка, интеграция, дизайн) и использует правильную терминологию. Главный недостаток, снижающий оценку по сравнению с предыдущим набором, — смешение

	проверяемо сть.	фокусировку на теме программных интерфейсов (API).	циклом (версионирова ние), безопасность, производитель ность, а также на современные подходы (gRPC, GraphQL, event-driven API).	development", "software integration" и "software interoperability "), так как они могут ссылаться на одни и те же работы.		контекстов. Включение "user interfaces" смещает фокус с программных интерфейсов (машина-маши на) на человеко-маши нные, что является отдельной, хотя и смежной, областью. Запросы вроде "engineering systems" без явной привязки к интерфейсам также слишком широки.
--	--------------------	--	---	--	--	---

Промпт 4						
DeepSeek	<b>Высокая.</b> Источники (arXiv) остаются академичес ки достоверны ми. Наличие обзоров (survey/revie w) является большим плюсом, так как такие работы систематизи руют знания и содержат множество проверяемы х ссылок на первичные источники.	<b>Высокая.</b> Запросы идеально отражают ключевые аспекты описания ВКР: Python API design, language binding, Python C extension, simulation automation. Они напрямую соответствуют цели работы — разработке языкового интерфейса на Python/C API для инженерного ПО.	<b>Средняя.</b> Запросы отлично покрывают техническую реализацию (Python/C API, связывание языков, расширения) и предметную область (многофазные потoki). Однако слабо представлены аспекты системного проектировани я: архитектурные паттерны для API, управление жизненным циклом модулей, интеграция со сторонними библиотеками	<b>Высокая.</b> Шесть фасетов охватывают четко разграниченны е аспекты: от общих принципов дизайна до конкретных технологий (C extensions) и предметной области (многофазные потoki). Пересечения между результатами по разным фасетам будут минимальным и содержательны ми.	<b>Высокая.</b> Простые web-safe запросы гарантируют стабильную и воспроизводим ую выдачу на arXiv. Стратегия добавления слов survey/review в отдельный запрос — отличная практика для поиска обобщающих материалов.	<b>Высокая.</b> Он демонстрирует глубокое понимание темы и её технических нюансов. Запросы сфокусирован ы именно на языковых интерфейсах, связывании кода и автоматизации инженерного ПО, а не на абстрактных веб-API.  Запрос multiphase flow simulation критически важен, так привязывает техническую задачу к реальной

			(NumPy, SciPy), которые упомянуты в описании как часть результата.			инженерной проблеме, что является сильной стороной работы.
ChatGPT	<b>Высокая.</b> Как и прежде, все источники — это научные препринты arXiv, имеющие идентификаторы, аннотации, авторов и четкую датировку, что обеспечивает полную проверяемость.	<b>Высокая.</b> Фасеты идеально соответствуют конкретной теме ВКР. Запросы точно соответствуют ключевым задачам работы: проектирование языковых API (Language API design), работа с Python/C API (Python C API), привязка к предметной области (Multiphase flow modeling), создание скриптуемых модулей (Scriptable modules Python) и их интеграция с научным стеком (Integrating scientific libraries).	<b>Высокая.</b> Это самый полный из представленных наборов. Он охватывает все ключевые аспекты ВКР: от общих принципов проектирования языковых интерфейсов до конкретной технологии реализации (C API), целевой инженерной области (моделирование потоков) и практических результатов работы (создание скриптуемых модулей, интеграция с NumPy/SciPy).	<b>Высокая.</b> Шесть фасетов четко разделены и покрывают независимые смысловые блоки работы. Пересечения между результатами по разным фасетам будут минимальны и содержательны (например, работа по Python API для CFD попадет в несколько фасетов, что правильно отражает её комплексность).	<b>Высокая.</b> Использование простых web-safe запросов гарантирует стабильные и воспроизводимые результаты на arXiv. Наличие review/survey запроса для каждого фасета — отличная практика для поиска обобщающих материалов.	<b>Очень высокая.</b> Он демонстрирует не просто понимание общей области, а глубокое погружение в структуру самой выпускной работы. Каждый фасет соответствует отдельному разделу анализа или этапу реализации.

4.1. Анализ сильных и слабых сторон моделей

Промпт 1: Простые поисковые запросы

Модель	Сильные стороны	Слабые стороны
<hr/>		

DeepSeek	Воспроизводимость и прозрачность, Разнообразие. Строго следует формату "только строки". Генерирует логически разные запросы, минимизируя семантическое пересечение.	Релевантность, Покрытие подтем. Понимает правила буквально, но теряет суть общей темы ("разработка API"). Запросы технически корректны, но не сфокусированы на цели.
ChatGPT	Релевантность. Лучше улавливает общую тему, включая в запросы больше ключевых терминов ("API", "interface", "design").	Разнообразие. Склонен к генерации семантически близких запросов, что ведет к дублированию результатов. Менее строг к уникальности каждой строки.

Вывод: Для такого промта DeepSeek надежнее технически, но ChatGPT дает более тематически релевантный набор, хотя и менее строгий.

## Промпт 2: Пакеты запросов (Basic + Soft-phrase)

Модель	Сильные стороны	Слабые стороны
DeepSeek	Достоверность, Разнообразие. Генерирует запросы, ведущие к конкретным программным инструментам и фреймворкам, что полезно для практика. Результаты четко разделены по областям.	Релевантность, Покрытие подтем. Фокус смещается на <i>результаты</i> разработки ПО (библиотеки), а не на <i>процесс</i> (проектирование API). Не хватает глубины темы.
ChatGPT	Релевантность (формальная). Активно использует заданные "якорные" слова в прямом значении.	Покрытие подтем, Разнообразие. Запросы часто касаются смежных областей (ML, UX), а не ядра темы. Пары запросов внутри пакета могут давать сильно пересекающиеся результаты.

Вывод: Обе модели справились средне. DeepSeek дал более разнообразный и технологичный, но узкий набор. ChatGPT пытался быть ближе к теме, но за счет смешения контекстов.

**Промпт 3: От общего к точному**

Модель	Сильные стороны	Слабые стороны
DeepSeek	Релевантность, Воспроизводимость, Итоговая оценка. Идеальное выполнение. Точно фокусируется на теме разработки API. Грамотно использует операторы <code>abs:</code> и <code>ti:</code> для точного поиска. Эскалация логична и эффективна. Технически безупречно.	Покрытие подтем (незначительное). Можно отметить отсутствие некоторых узких подтем (версионирование, безопасность), но в рамках 10 наборов это допустимо.
ChatGPT	Воспроизводимость. Также корректно применяет синтаксис расширенного поиска. Часть запросов высоко релевантна.	Релевантность, Разнообразие. Критическая ошибка — смешение понятий "программный интерфейс (API)" и "пользовательский интерфейс (UI)". Это размывает фокус. Запросы иногда слишком общие.

Вывод: DeepSeek явно превосходит в работе со сложными, структурированными промтами, требующими технической точности и сохранения фокуса. Здесь его буквальность стала преимуществом.

**Промпт 4: Фасетный поиск на основе текста**

Модель	Сильные стороны	Слабые стороны
--------	-----------------	----------------

DeepSeek	Достоверность, Релевантность, Воспроизводимость, Итоговая оценка. Глубоко понимает технические нюансы исходного текста. Запросы точно отражают ключевые технологии (Python/C API, binding). Отлично привязан к предметной области (многофазные потоки).	Покрытие подтем. Недостаточно освещены аспекты интеграции со сторонними библиотеками и системного дизайна API, хотя они упомянуты в тексте.
ChatGPT	Релевантность, Покрытие подтем, Итоговая оценка. Превосходное выполнение. Демонстрирует исключительное семантическое понимание. Выделенные фасеты идеально соответствуют структуре и содержанию описанной работы. Охватывает тему максимально полно, включая все ключевые аспекты.	Практически отсутствуют. Можно отметить лишь чуть меньшую техническую "остроту" в формулировках по сравнению с DeepSeek.

Вывод: Обе модели справились блестяще, но ChatGPT показал образцовое понимание контекста. Он не просто сгенерировал запросы, а реконструировал логику и структуру исходной работы, что является высшим пилотажем в анализе текста. Здесь его способность к семантическому синтезу проявилась в полной мере.

## 5. Разработка собственного промпта (Оптимизация).

На основе проведенного сравнения (Таблица 1), модель ChatGPT — она показывает лучшее семантическое понимание, более широкое и релевантное покрытие тем, а также более гибко адаптируется к сложным промптам. DeepSeek технически строг, но слишком буквальный и узко фокусируется, тогда как ChatGPT чаще сохраняет смысл и контекст.

Основные проблемы существующих промптов:

1. **Смещение контекстов:** Запросы часто уходят в смежные области (UI/UX, ML-фреймворки), а не фокусируются на разработке программных интерфейсов (API) как инженерной дисциплине.
2. **Поверхностное покрытие:** Не хватает запросов на ключевые инженерные аспекты (версионирование, безопасность, производительность, стандарты вроде OpenAPI/gRPC).
3. **Слабая привязка к домену:** Недостаточно запросов, связывающих технику (Python/C API, биндинги) с конкретной инженерной областью (моделирование, CAD, симуляции).
4. **Низкая воспроизводимость:** Простые запросы без операторов дают слишком шумные результаты.

**Улучшенный промпт** сочетает сильные стороны Промпта 3 (эскалация, точность за счет полей `ti:/abs:`) и Промпта 4 (фасетный, системный подход, привязка к теме ВКР). Он исправляет выявленные недостатки, фокусируясь строго на разработке программных интерфейсов для инженерного ПО.

Контекст: Тема — «Разработка программных интерфейсов в современных инженерных приложениях». Фокус на проектировании, реализации и интеграции языковых API (особенно на Python/C API) для научно-инженерного ПО (моделирование, симуляции, CAD, CAE), включая автоматизацию, расширяемость и связывание со сторонними библиотеками (NumPy, SciPy).

Задача: Сгенерируй 8 наборов эскалируемых запросов для arXiv. Каждый набор должен глубоко раскрывать отдельный критический аспект темы. Используй следующий обязательный формат для каждого набора из 3-х строк:

1. Q0 (Общий, web-safe): 3-5 ключевых слов. Должен содержать ровно один из этих «якорных» терминов: API, interface, binding, extension. *Пример:*

engineering simulation Python API.

2. Q1 (Уточненный, web-safe+): Одна короткая фраза в кавычках (2-3 слова) + 1-2 общих слова. Фраза должна быть точным термином из области разработки ПО или инженерии. *Пример:* "API design" engineering software.
3. Q2 (Целевой, для Advanced Search): abs:"<фраза до 3 слов>" OR ti:"<фраза до 3 слов>". Используй ту же или более конкретную фразу, что в Q1. Не добавляй NOT, AND, OR кроме указанного, категории (cat:) или другие поля.

Жесткие правила и фокус:

- Язык: Только английский.
- Запрещено: Редкие узкие аббревиатуры (кроме API), NOT/AND/OR (кроме OR в Q2), поля кроме abs: и ti: в Q2, двоеточия вне операторов Q2, длинные точные фразы (>3 слов в кавычках).
- Фокус на инженерное ПО: Связывай запросы с доменами: simulation, modeling, CAD, CAE, engineering, scientific computing.
- Охват аспектов (каждый набор — один аспект):
  1. Принципы проектирования: Архитектура, паттерны, контракты.
  2. Технологии связывания и расширения: Python/C API, биндинги, расширения (extensions).
  3. Интеграция и автоматизация: Связь с инженерными системами, скриптинг, автоматизация рабочих процессов.
  4. Производительность и оптимизация: Вычислительная эффективность, ускорение, низкоуровневые взаимодействия.
  5. Управление жизненным циклом и качество: Версионирование, тестирование, документация, депривация.
  6. Стандарты и протоколы: REST, gRPC, GraphQL, OpenAPI в контексте инженерных систем.
  7. Предметно-ориентированные API (DSL): API для конкретных областей (CFD, многофазные потоки, конечно-элементный анализ).
  8. Интеграция научных библиотек: Связь с NumPy, SciPy, Matplotlib в инженерных конвейерах.

Формат вывода: Выведи ровно 24 строки (8 наборов \* 3 строки). Каждый набор — три строки подряд: сначала Q0, затем Q1, затем Q2. Не добавляй нумерацию, пояснения или дополнительные символы.

Начало вывода (пример для первого аспекта):

engineering software API design principles

"API design" software engineering

abs:"API design" OR ti:"API design"



## 5.1. Применение промпта.

Ответ:

engineering API architecture patterns  
"interface architecture" engineering software  
abs:"interface architecture" OR ti:"interface architecture"  
binding technologies Python/C API  
"Python/C API" scientific computing  
abs:"Python/C API" OR ti:"Python/C API"  
engineering interface automation scripting  
"interface automation" engineering workflows  
abs:"interface automation" OR ti:"interface automation"  
performance API optimization  
"API optimization" high performance  
abs:"API optimization" OR ti:"API optimization"  
software interface quality testing  
"interface testing" software quality  
abs:"interface testing" OR ti:"interface testing"  
standards API protocols  
"API protocols" engineering systems  
abs:"API protocols" OR ti:"API protocols"  
domain-specific API CFD  
"API CFD" domain modeling  
abs:"API CFD" OR ti:"API CFD"  
bindings with NumPy SciPy  
"NumPy bindings" scientific pipelines  
abs:"NumPy bindings" OR ti:"NumPy bindings"

## 5.2. Оценка финального ответа.

Достоверно сть и проверяемо сть источников	Релевантнос ть теме	Покрытие подтем (полнота)	Разнообрази е без дублей	Воспроизво димось и прозрачнос ть	Итоговая оценка
<b>Высокая.</b> Все запросы	<b>Очень высокая.</b>	<b>Высокая.</b> Набор	<b>Высокая.</b> Каждый из 8	<b>Очень высокая.</b>	<b>Очень высокая.</b>

нацелены на arXiv, репозиторий академических препринтов с высоким уровнем достоверности и полной проверяемостью по идентификаторам.	Запросы не просто повторяют тему, а глубоко раскрывают её критические инженерные аспекты: архитектура, производительность, тестирование, интеграция с научным стеком (NumPy), применение в предметной области (CFD). Это именно те вопросы, с которыми сталкивается разработчик при создании промышленного API.	покрывает ключевые аспекты: от выбора технологии связывания (Python/C API) и архитектуры до оптимизации, тестирования и интеграции. Запрос "API CFD" напрямую связывает тему с целевой предметной областью (моделирование потоков).	наборов раскрывает уникальный и важный аспект: архитектура, технологии, автоматизация, производительность, тестирование, стандарты, предметная область, интеграция. Пересечения будут минимальными и осмысленными.	Четкое следование формату с использованием операторов поля (abs., ti:) делает поиск максимально точным и полностью воспроизводимым. Это лучшая практика для arXiv.	Набор запросов переводит поиск из плоскости общих тем в плоскость конкретных инженерных проблем. Такой подход позволяет найти не просто статьи "про API", а работы, решающие схожие технические задачи: как оптимизировать вызовы, как тестировать интерфейсы, как строить архитектуру для научного ПО.
---	---	---	--	--	---

### Ключевые результаты оптимизации:

Оптимизация привела к созданию профессионального инструмента для академического поиска. Сгенерированные запросы позволяют находить не просто статьи "об API", а научные работы, предлагающие решения конкретных технических задач, с которыми сталкивается разработчик при создании языковых интерфейсов для сложного инженерного программного обеспечения. Это напрямую соответствует цели теоретического анализа в рамках ВКР.

## 6. Выводы

Проведенный сравнительный анализ ответов моделей DeepSeek и ChatGPT на серию специализированных промптов для формирования поисковых запросов к arXiv позволил выявить их сильные и слабые стороны, а также разработать оптимизированный промпт.

Ключевые выводы по моделям:

- DeepSeek демонстрирует высокую техническую точность, строгое следование формату и хорошую воспроизводимость результатов. Однако модель склонна к буквальному пониманию задач, что иногда приводит к потере общего контекста и недостаточной фокусировке на сути темы.
- ChatGPT проявляет более глубокое семантическое понимание, лучше улавливает тематическую релевантность и демонстрирует способность к комплексному охвату различных аспектов темы. Особенно сильной стороной является работа с контекстно-зависимыми промптами, где модель успешно реконструирует логику исходного материала.

Таким образом, оптимальная стратегия заключается в использовании ChatGPT для анализа контекста и выделения смысловых аспектов темы, с последующей технической доработкой и формализацией запросов по образцу, демонстрируемому DeepSeek. Разработанный финальный промпт успешно интегрирует сильные стороны обеих моделей, создавая эффективный инструмент для академического поиска по специализированной инженерной тематике.