

Лабораторная работа по дисциплине «Автоматизация научных исследований»

Соломатов Александр Денисович

Гр. 5040102/50201

December 2025

1 Цель работы

Сформулировать запросы к системе поиска научной литературы для нахождения релевантных работ, относительно приведенного научного текста. Для поиска использовать **arXiv** (<https://arxiv.org/search/>).

2 Задание

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 четырёх предложенных промтов.
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 промтов, составить собственный промт, который, по вашему мнению, должен дать наилучший результат по всем критериям.
6. Финальная проверка. Прогнать статью через выбранную модель, используя свой авторский промт.

3 Критерии оценки

- Достоверность и проверяемость источников

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

4 Промты

4.1 Промт 1

Промт 1 (простые поисковые запросы)

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

Требования:

- Английский язык, -24 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: DSL GENERATION, LLM PYTHON CODEGEN, EBNF, VISITOR PATTERN, METAMODELING, AST TRANSLATION, IDE INTEGRATION - выбери подходящие к теме.
- Выведи только готовые строки запросов, по одному на строку.

Оформи все в виде .tex тех кода без ссылок на приложенную статью

Используй itemize.

4.2 Промт 2

Промт 2 (пакеты запросов)

Подготовь 10 пакетов запросов для сайта arXiv (<https://arxiv.org/search>).

Каждый пакет - два web-safe варианта, которые дают выдачу:

- Basic: -36 ключевых слов без кавычек и операторов только(пробелы).
- Soft-phrase: одна короткая фраза в кавычках до 3 слов + -12 общих слова.

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 (DSL GENERATION, LLM PYTHON CODEGEN, EBNF, VISITOR PATTERN, METAMODELING, AST TRANSLATION, IDE INTEGRATION).

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

Оформи все в виде .tex тех кода без ссылок на приложенную статью

Используй itemize для внешней нумерации и enumerate для внутренней

4.3 Промт 3

Промт 3 (эскалируемые запросы)

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

- Q0 (web-safe): -35 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) + -12 общих слова. Без операторов.
- Q2 (fielded мягкий, для API или Advanced Search): аббревиатура:"< фраза до 3 слов>" OR тикратка:"< фраза до 3 слов>" - без NOT; категорию не добавляй.

Правила:

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

Оформи все в виде .tex тех кода без ссылок на приложенную статью

Используй itemize для внешней нумерации и enumerate для внутренней

4.4 Промт 4

Промт 4 (фасеты и запросы)

Выдели 6 фасетов подтем() по тексту ниже каждый(-23 слова), но не выводи их отдельно.

Для каждого фасета сгенерируй по 3 запроса, предназначенные для arXiv и дающие выдачу:

- Minimal (web-safe): -24 общих ключевых слова, без кавычек и операторов.
- Broadened (web-safe): добавь -12 синонима через пробел без(OR), оставь якорные слова (DSL GENERATION, LLM PYTHON CODEGEN, EBNF, VISITOR PATTERN, METAMODELING, AST TRANSLATION, IDE INTEGRATION).

- 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.

Оформь все в виде .tex тех кода без ссылок на приложенную статью

Используй itemize

4.5 Оптимизированный Промт 5

Промт 5 (Гибридный прецизионный поиск)

Сгенерируй 8 наборов высокорелевантных поисковых запросов для arXiv.org по теме автоматизации языкового инструментария и LLMкодогенерации-.

Для каждого набора выведи три уровня сложности:

1. Level A (Contextual Keywords): -34 слова без кавычек. Объединяй «якорный» термин с контекстным словом например(, не просто «visitor »pattern, а «visitor pattern »compiler). Это минимизирует шум.

2. Level B (Qualified Phrase): Короткая фраза в кавычках (2-3 слова) + 1 уточняющее слово вне кавычек. Ориентировано на захват конкретных концепций.

3. Level C (Advanced Boolean): Строгий запрос для Advanced Search с использованием полей титла и аннотации. Формат: (tiфраза:"" OR absфраза:"") AND контекстноеслово_.

Требования:

- Якорные термины обязательно(использовать все в разных наборах): DSL GENERATION, LLM PYTHON CODEGEN, EBNF, VISITOR PATTERN, METAMODELING, AST TRANSLATION, IDE INTEGRATION.

- Только английский язык.

- Избегай «галлюцинаций» в синтаксисе: не используй операторы, которые не

поддерживает arXiv например(, NEAR или NOT в простых строках).

Формат вывода:

Itemize для наборов, Enumerate (A, B, C) для уровней внутри. Без вводных фраз.

5 Результаты

5.1 Промт 1

Gemini

- DSL generation automation
- LLM python codegen evaluation
- EBNF grammar parser
- visitor pattern design
- metamodeling software engineering
- AST translation deep learning
- IDE integration tool
- neural DSL generation
- transformer LLM python codegen
- metamodeling domain specific
- EBNF syntax analysis
- AST translation source code

DeepSeek

- DSL GENERATION Python
- VISITOR PATTERN compiler
- AST TRANSLATION code
- METAMODELING framework
- LLM PYTHON CODEGEN

- EBNF grammar language
- IDE INTEGRATION plugin
- DSL GENERATION tools
- METAMODELING eclipse
- AST TRANSLATION transformation
- VISITOR PATTERN
- LLM code generation

5.2 Проект 2

Gemini

- **Packet 1: Domain Specific Languages**
 1. dsl generation software language development automation
 2. "dsl generation" framework approach
- **Packet 2: Neural Code Generation**
 1. llm python codegen machine learning models
 2. "llm python codegen" transformer evaluation
- **Packet 3: Formal Grammars**
 1. ebnf grammar syntax parsing formal specification
 2. "ebnf" language definition
- **Packet 4: Software Design Patterns**
 1. visitor pattern design principles compiler implementation
 2. "visitor pattern" object oriented
- **Packet 5: Model-Driven Engineering**
 1. metamodeling architecture model driven systems design
 2. "metamodeling" domain semantics
- **Packet 6: Compiler Trees**
 1. ast translation source code abstract syntax
 2. "ast translation" mapping algorithms

- **Packet 7: Developer Environment**

1. ide integration developer tools environment productivity
2. "ide integration" plugin architecture

- **Packet 8: Generative Programming**

1. dsl generation compiler construction code optimization
2. "dsl generation" synthesis tools

- **Packet 9: Automated Programming**

1. llm python codegen neural code benchmarks
2. "llm python codegen" deep learning

- **Packet 10: Syntax Transformation**

1. ast translation program transformation static analysis
2. "ast translation" intermediate representation

DeepSeek

- **Packet 1**

1. DSL GENERATION parser tools lexical syntax grammar
2. "domain specific language" METAMODELING generation tools

- **Packet 2**

1. LLM PYTHON CODEGEN code generation parsing large models
2. "code generation" AST TRANSLATION large language models

- **Packet 3**

1. EBNF grammar compiler syntax parser design
2. "extended BNF" IDE INTEGRATION grammar tools

- **Packet 4**

1. VISITOR PATTERN compiler design traversal algorithms
2. "visitor pattern" AST TRANSLATION compiler implementation

- **Packet 5**

1. METAMODELING domain modeling language workbench
2. "meta modeling" DSL GENERATION language engineering

- **Packet 6**

1. AST TRANSLATION intermediate representation compiler passes
2. "abstract syntax" VISITOR PATTERN translation tools

- **Packet 7**

1. IDE INTEGRATION language server protocol editor tools
2. "IDE integration" EBNF grammar support plugins

- **Packet 8**

1. DSL GENERATION language workbench visual programming
2. "visual languages" METAMODELING development environment

- **Packet 9**

1. LLM PYTHON CODEGEN program synthesis neural networks
2. "neural code generation" AST TRANSLATION python

- **Packet 10**

1. EBNF parser generator compiler construction syntax
2. "parser generator" IDE INTEGRATION development tools

5.3 **Промт 3**

Gemini

- **Translator Building Systems**

1. Q0: translator building systems compilers
2. Q1: "translator building" tools
3. Q2: abs:"translator building" OR ti:"translator building"

- **Parsing Expression Grammars (PEG)**

1. Q0: parsing expression grammar rust
2. Q1: "parsing expression grammar" peg
3. Q2: abs:"parsing expression" OR ti:"parsing expression"

- **Context-Free Grammar Regularization**

1. Q0: context free grammar regularization
2. Q1: "context free grammar" transformation

3. Q2: abs:"grammar regularization" OR ti:"grammar regularization"

- **Domain-Specific Languages (DSL)**

1. Q0: domain specific language modeling
2. Q1: "domain specific" language
3. Q2: abs:"domain specific" OR ti:"domain specific"

- **Parser Generator Algorithms**

1. Q0: parser generator tools development
2. Q1: "parser generator" systems
3. Q2: abs:"parser generator" OR ti:"parser generator"

- **LLM Code Generation**

1. Q0: llm code generation transformers
2. Q1: "llm code generation" research
3. Q2: abs:"llm code generation" OR ti:"llm code generation"

- **Visual Programming and MetaCASE**

1. Q0: visual programming metacase tools
2. Q1: "visual programming" environments
3. Q2: abs:"visual programming" OR ti:"visual programming"

- **Multi-Level Modeling**

1. Q0: multi level modeling software
2. Q1: "multi level" modeling
3. Q2: abs:"multi level modeling" OR ti:"multi level modeling"

- **LLM-based Programming Assistance**

1. Q0: large language models programming
2. Q1: "large language models" code
3. Q2: abs:"llm programming" OR ti:"llm programming"

- **Automated Code Synthesis**

1. Q0: grammar based code synthesis
2. Q1: "code synthesis" automation
3. Q2: abs:"code synthesis" OR ti:"code synthesis"

DeepSeek

- 1. Q0: parsing expression grammar PEG compiler
2. Q1: "parsing expression grammar" tools
3. Q2: abs:"parsing expression grammar" OR ti:"parsing expression grammar"
- 1. Q0: domain specific language visual modeling
2. Q1: "domain specific" language modeling
3. Q2: abs:"domain specific language" OR ti:"domain specific language"
- 1. Q0: context free grammar regularization
2. Q1: "context free" grammar regularization
3. Q2: abs:"context free grammar" OR ti:"context free grammar"
- 1. Q0: parser generator LALR LL
2. Q1: "parser generator" compiler
3. Q2: abs:"parser generator" OR ti:"parser generator"
- 1. Q0: multi level modeling metamodeling
2. Q1: "multi level" modeling environment
3. Q2: abs:"multi level modeling" OR ti:"multi level modeling"
- 1. Q0: syntax graph transformation tools
2. Q1: "graph transformation" syntax
3. Q2: abs:"graph transformation" OR ti:"graph transformation"
- 1. Q0: visual language development tools
2. Q1: "visual language" development
3. Q2: abs:"visual language" OR ti:"visual language"
- 1. Q0: attribute grammar semantics processing
2. Q1: "attribute grammar" semantics
3. Q2: abs:"attribute grammar" OR ti:"attribute grammar"
- 1. Q0: translator building systems compiler
2. Q1: "translator building" systems
3. Q2: abs:"translator building" OR ti:"translator building"
- 1. Q0: formal grammar transformation tools
2. Q1: "grammar transformation" tools
3. Q2: abs:"grammar transformation" OR ti:"grammar transformation"

5.4 Проект 4

Gemini

- python code generation
- python code generation llm python codegen transformer
- python code generation survey
- formal grammar specification
- formal grammar specification ebnf syntax
- formal grammar specification review
- compiler design patterns
- compiler design patterns visitor pattern traversal
- compiler design patterns survey
- domain model engineering
- domain model engineering metamodeling abstraction
- domain model engineering review
- abstract syntax transformation
- abstract syntax transformation ast translation tree
- abstract syntax transformation survey
- language synthesis tools
- language synthesis tools dsl generation ide integration
- language synthesis tools review

DeepSeek

- compiler construction tools
- compiler DSL GENERATION code generation
- compiler construction review survey
- grammar transformation regularization
- grammar transformation CFG EBNF

- grammar transformation survey
- parser generator ANTLR
- parser LLM PYTHON CODEGEN Bison
- parser generator review survey
- PEG parsing expressions
- PEG parsing AST TRANSLATION
- PEG parsing survey review
- visual modeling languages
- visual METAMODELING domain languages
- visual modeling languages review
- IDE language plugin
- IDE INTEGRATION language tools
- IDE language plugin survey

5.5 Проект 5

Gemini

- **Set 1: DSL Generation**
 1. DSL generation automated synthesis
 2. "DSL generation" transformer
 3. (ti:"DSL generation" OR abs:"DSL generation") AND framework
- **Set 2: LLM Python CodeGen**
 1. LLM Python codegen benchmark
 2. "Python code generation" LLM
 3. (ti:"Python code generation" OR abs:"Python code generation") AND transformer
- **Set 3: EBNF Grammar Engineering**
 1. EBNF grammar inference LLM
 2. "EBNF grammar" parser

3. (ti:"EBNF" OR abs:"EBNF") AND learning
- **Set 4: Visitor Pattern Automation**
 1. visitor pattern compiler optimization
 2. "visitor pattern" boilerplate
 3. (ti:"visitor pattern" OR abs:"visitor pattern") AND automated
 - **Set 5: Metamodeling and Synthesis**
 1. metamodeling software engineering LLM
 2. "metamodeling approach" synthesis
 3. (ti:"metamodeling" OR abs:"metamodeling") AND code
 - **Set 6: AST Translation Techniques**
 1. AST translation code migration
 2. "AST translation" neural
 3. (ti:"abstract syntax tree" OR abs:"abstract syntax tree") AND translation
 - **Set 7: IDE Integration Systems**
 1. IDE integration developer productivity
 2. "IDE integration" tool
 3. (ti:"IDE integration" OR abs:"IDE integration") AND LLM
 - **Set 8: Advanced Program Synthesis**
 1. program synthesis language tooling
 2. "program synthesis" benchmark
 3. (ti:"program synthesis" OR abs:"program synthesis") AND LLM

6 Анализ результатов

В данном разделе приводится экспертная оценка поисковых параметров, предложенных моделями Gemini и DeepSeek. Анализ сфокусирован на применимости данных для проведения систематического обзора литературы в области автоматизации разработки языков программирования и нейронной генерации кода через интерфейс arXiv.org.

6.1 Оценки

Оценка производилась по 10-балльной шкале на основе следующих критериев:

- (a) Достоверность и проверяемость;
- (b) Релевантность теме;
- (c) Покрытие подтем (полнота);
- (d) Разнообразие без дублей;
- (e) Воспроизводимость и прозрачность;
- (f) Техническая применимость (соответствие синтаксису поиска arXiv).

Модель / Промпт	(a)	(b)	(c)	(d)	(e)	(f)	Среднее
Gemini (П1)	9.0	9.0	7.0	8.0	8.5	7.0	8.08
DeepSeek (П1)	8.0	8.0	7.0	7.0	8.0	6.0	7.33
Gemini (П2)	9.0	8.0	9.0	9.0	8.5	8.0	8.58
DeepSeek (П2)	8.0	9.0	8.0	8.0	8.0	7.0	8.00
Gemini (П3)	10.0	10.0	9.0	9.0	9.0	10.0	9.50
DeepSeek (П3)	10.0	9.0	9.0	9.0	8.5	10.0	9.25
Gemini (П4)	9.0	9.0	10.0	9.0	9.5	9.0	9.25
DeepSeek (П4)	9.0	8.0	8.0	9.0	9.0	8.0	8.50
Gemini (П5)*	10.0	10.0	10.0	9.0	10.0	10.0	9.83

Table 1: Сводная таблица оценок. Промпт 5 — оптимизированный авторский вариант.

6.2 Общий анализ

На основе анализа представленных данных выявлены следующие ключевые проблемы и сильные стороны использования данных параметров для поиска:

- **Проблема семантической избыточности (Промт 1):** Использование простых терминов вроде «*visitor pattern*» или «*metamodeling*» без контекста (например, без привязки к конкретному ПО или теории) на arXiv приводит к огромному количеству шума. Модели склонны выдавать общеинженерные термины, которые в академической среде часто рассматриваются в контексте более узких дисциплин.
- **Специфика синтаксиса arXiv (Промт 2 и 3):** Промт 3 является наиболее релевантным для системного анализа, так как использует полевой поиск (`abs:`, `ti:`). Однако выявлена слабая сторона: модели иногда заключают в кавычки слишком длинные фразы (например, «*context free grammar regularization*»), что в поисковом движке arXiv может привести к нулевой выдаче из-за избыточной специфичности.

- **Полнота и фасетный поиск (Промт 4):** Использование фасетов (Minimal, Broadened, Review-oriented) — наиболее эффективная стратегия. Gemini показала лучшую способность к генерации запросов типа «survey» и «review», что критически важно для аналитика при поиске фундаментальных работ. DeepSeek продемонстрировал склонность к использованию верхнего регистра (например, *DSL GENERATION*), что не влияет на технический результат, но снижает читаемость запросов.
- **Терминологическая точность:** Обе модели корректно идентифицировали «якорные» темы: *AST translation*, *EBNF grammar* и *LLM python codegen*. Особо стоит отметить Промт 3 у Gemini, где предложены запросы по *PEG (Parsing Expression Grammars)* — это крайне релевантная и современная альтернатива классическим грамматикам в актуальных исследованиях.

Заключение: Наиболее надежным методом поиска для системного аналитика является использование структуры запросов из **Промта 3**, так как они минимизируют ложноположительные результаты за счет таргетирования заголовков и аннотаций.

6.3 Вывод по оптимизированному промπτу

Использование оптимизированного промпта №5 позволило достичь наиболее высоких показателей по всем метрикам системного анализа. Основные качественные улучшения включают:

- **Ликвидация семантического шума:** Благодаря введению **Level A (Contextual Keywords)**, модель перестала генерировать изолированные термины. Например, в *Set 4* термин *visitor pattern* жестко связан с *compiler optimization*, что исключает попадание в выдачу статей по общему проектированию.
- **Техническая точность Level C:** В отличие от Промпта 3, где сложные фразы часто приводили к пустой выдаче, Промпт 5 использует гибридную логику: **(ti:"фраза" OR abs:"фраза") AND слово**. Это гарантирует наличие результата даже при отсутствии точного совпадения длинной фразы в заголовке, расширяя поиск через аннотацию.
- **Полнота охвата:** Модель успешно покрыла все 8 целевых доменов, включая такие специфические темы, как *EBNF Grammar Engineering* и *AST Translation*, обеспечив при этом разнообразие подходов (от бенчмарков до нейронных методов).

Общий итог: Работа подтвердила, что качество результата существенно зависит как от выбранной модели (Gemini продемонстрировала более высокую надежность и стабильное соблюдение формальных требований), так и от точности формулировки промпта. Четкие, структурированные инструкции с явными ограничениями (внедренные в авторском Промпте 5) минимизируют технические ошибки и существенно повышают полезность сгенерированных поисковых запросов для автоматизации научного исследования.