ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Магистерская программа
«Исследования и предпринимательство
в искусственном интеллекте»

СОГЛАСОВАНО

Научный руководитель,
доцент ФКН департамента
программной инженерии, к.т.н.

_____ С. Л. Макаров
«_____» _____ 2025 г.

УТВЕРЖДАЮ

Академический руководитель
образовательной программы
«Исследования и предпринимательство
в искусственном интеллекте»

_____ Д. С. Лялин
«_____» _____ 2025 г.

Курсовая работа
(проектная)

на тему: **Разработка мультиагентной текстовой стратегической игры на основе оркестрируемых языковых моделей**

по направлению подготовки 01.04.02 «Прикладная математика и информатика»

ВЫПОЛНИЛ

студент группы ИПИИ
образовательной программы
01.04.02 «Прикладная
математика и информатика»

_____ Н. С. Пеганов
«_____» _____ 2025 г.

Москва 2025

Реферат

Работа посвящена разработке мультиагентной текстовой стратегической игры на основе оркестрируемых языковых моделей и исследованию применимости современных LLM в качестве автоматизированного мастера игры в жанре военно-политических игр (ВПИ).

В работе рассмотрен полный цикл создания подобной системы, включая проектирование архитектуры мультиагентной среды, оркестрацию языковых моделей для выполнения различных игровых функций, а также анализ результатов пилотного запуска с реальными пользователями. Особое внимание уделяется методам преодоления типичных проблем языковых моделей в контексте игрового процесса: галлюцинации, сохранение долгосрочного контекста и вычислительные ограничения.

Результаты исследования демонстрируют потенциал современных LLM для автоматизации роли вердера в текстовых стратегических играх, выявляют ключевые технические и игровые ограничения существующих подходов, а также предлагают набор технических решений для повышения качества игрового опыта, включая применение систем на основе RAG и локальных моделей для снижения стоимости эксплуатации игровой системы.

Данная работа состоит из 11 страниц, 2 глав, 5 листингов, 1 таблицы, 2 приложений. Использовано 6 источников.

Ключевые слова: мультиагентная система; языковые модели; оркестрация LLM; текстовые стратегические игры; искусственный интеллект в играх; RAG; автоматизация игрового мастера.

Abstract

This paper is dedicated to developing a multi-agent text-based strategy game using orchestrated large language models, and investigating the applicability of modern LLMs as automated game masters in the military-political games (WPG) genre.

The work covers the complete cycle of creating such a system, including designing the architecture of a multi-agent environment, orchestrating language models to perform various game functions, and analyzing the results of a pilot launch with real users. Special attention is paid to methods for overcoming typical language model problems in the gaming context: hallucinations, maintaining long-term context, and computational limitations.

The research results demonstrate the potential of modern LLMs for automating the role of game arbiter in text-based strategy games, identify key technical and gameplay limitations of existing approaches, and propose a set of technical solutions to enhance gaming experience, including the application of RAG-based systems and local models to reduce the operational costs of the game system.

The paper contains 11 pages, 2 chapters, 5 listings, 1 table, 2 appendices. 6 sources are used.

Keywords: multi-agent system; large language models; LLM orchestration; text-based strategy games; artificial intelligence in games; RAG; automated game mastering.

Содержание

Используемые определения и термины

Common Vulnerabilities and Exposures (CVE) – база данных общеизвестных уязвимостей информационной безопасности.

Common Weakness Enumeration (CWE) – общий перечень и система классификации слабых мест и уязвимостей программного обеспечения.

Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems.

Kotlin – статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и разрабатываемый компанией JetBrains.

Абстрактное синтаксическое дерево (АСД, Abstract Syntax Tree, AST) – одна из форм промежуточного представления программ в виде древовидной структуры.

Анализ потока данных (Data Flow Analysis, DFA) – один из основных методов анализа программ, позволяющий определить в каждой точке программы некоторую информацию о данных, которыми оперирует код.

Байткод – одна из форм промежуточного представления программ в виде инструкций, которые близки к машинным и могут быть интерпретированы при помощи виртуальной машины.

Виртуальная машина Java (Java Virtual Machine, JVM) – основная часть исполняющей системы Java, исполняющая байткод, полученный из исходного кода программы, на конкретной платформе путём трансляции байткода в машинные инструкции.

Граф потока управления (ГПУ, Control Flow Graph, CFG) – множество всех возможных путей выполнения программы, представленное в виде графа.

Промежуточное представление (Intermediate Representation, IR) – структура данных или код, используемый внутри компилятора или виртуальной машины для представления программ.

Статический анализ кода – анализ исходного кода на предмет ошибок и недочётов без непосредственного выполнения анализируемых программ.

Введение

Пример введения.

Это пример ссылки на статью [1].

А это пример ссылки на онлайн-ресурс [2].

А это пример нескольких ссылок [3—6].

Глава 1. Обзор источников

Текст главы 1

1.1. Какая-то подглава

Текст подглавы

1.1.1. Какая-то подподглава

Текст подподглавы

1.1.1.1. Какой-то параграф

Текст параграфа

1.1.1.2. Какой-то параграф

Текст параграфа

1.1.1.3. Какой-то параграф

Текст параграфа

1.1.1.4. Какой-то параграф

Текст параграфа

1.1.2. Какая-то подподглава

Текст подподглавы

1.1.2.1. Какой-то параграф

Текст параграфа

1.1.2.2. Какой-то параграф

Текст параграфа

1.1.2.3. Какой-то параграф

Текст параграфа

1.1.2.4. Какой-то параграф

Текст параграфа

Выводы по главе

Текст Текст Текст Текст Текст Текст Текст

Глава 2. Какая-нибудь ещё глава

Текст главы 2

Заключение

Текст заключения

Список использованных источников

1. *Shelekhov V. I.*, *Kuksenko S. V.* Data flow analysis of Java programs in the presence of exceptions // International Andrei Ershov Memorial Conference on Perspectives of System Informatics. — Springer. 1999. — С. 389—395.
2. Common Weakness Enumeration [Электронный ресурс] : CWE-703: Improper Check or Handling of Exceptional Conditions. — URL: `https://cwe.mitre.org/data/definitions/703.html` (дата обр. 31.12.2021).
3. SonarRules [Электронный ресурс] : Java static code analysis. — URL: `https://rules.sonarsource.com/java/tag/error-handling/` (дата обр. 22.04.2022).
4. SpotBugs [Электронный ресурс] : Bug descriptions. — URL: `https://spotbugs.readthedocs.io/en/latest/bugDescriptions.html` (дата обр. 22.04.2022).
5. Infer [Электронный ресурс] : List of all issue types. — URL: `https://fbinfer.com/docs/all-issue-types/` (дата обр. 22.04.2022).
6. Detekt [Электронный ресурс] : Exceptions Rule Set. — URL: `https://detekt.dev/exceptions.html` (дата обр. 22.04.2022).

Пример приложения

Пример приложения. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст. Какой-то текст.

Ссылка на приложение Б.

Тут ссылка на листинг 1.

А тут ссылка на листинг 3.

```
@Deprecated("Reason")
fun findScriptDefinition(project: Project, script: SourceCode): ScriptDefinition?
    {
  val scriptDefinitionProvider = ScriptDefinitionProvider.getInstance(project) ?:
      return null
  ?: throw IllegalStateException("Unable to get script definition: ...")

  return scriptDefinitionProvider.findDefinition(script) ?:
      scriptDefinitionProvider.getDefaultDefinition() // Comment
}
```

Листинг 1 — Пример какого-то кода на Kotlin

```
class Main {
  public static ScriptDefinition findScriptDefinition(Project project, SourceCode
      script) {
    ScriptDefinitionProvider scriptDefinitionProvider = ScriptDefinitionProvider.
        getInstance(project);
    if (scriptDefinitionProvider == null) {
      if (null == null) {
        throw IllegalStateException("Unable to get script definition: ...");
      } else {
        return null;
      }
    }

    ScriptDefinition definition = scriptDefinitionProvider.findDefinition(script);
    if (definition == null) {
      return scriptDefinitionProvider.getDefaultDefinition(); // Comment
    } else {
      return definition;
    }
  }
}
```

Листинг 2 — Пример какого-то кода на Java

```
...
13 aload_2
14 dup
15 ifnonnull    28
18 new          #17 // NullPointerException
21 dup
22 ldc          #19 // String null cannot be cast to non-null String
24 invokespecial #23 // NullPointerException."<init>"(String)
27 athrow
...
46 aload_2
47 dup
48 ifnonnull    61
51 new          #17 // NullPointerException
54 dup
55 ldc          #19 // String null cannot be cast to non-null String
57 invokespecial #23 // NullPointerException."<init>"(String)
60 athrow
...
```

Листинг 3 — Пример JVM-байткода

```
...
13: aload_2
14: dup
15: ifnonnull    28
18: new          #17 // NullPointerException
21: dup
22: ldc          #19 // String null cannot be cast to non-null String
24: invokespecial #23 // NullPointerException."<init>"(String)
27: athrow
...
46: aload_2
47: dup
48: ifnonnull    61
51: new          #17 // NullPointerException
54: dup
55: ldc          #19 // String null cannot be cast to non-null String
57: invokespecial #23 // NullPointerException."<init>"(String)
60: athrow
...
```

Листинг 4 — Пример JVM-байткода 2

А тут ссылка на таблицу 1.

Таблица 1 — Пример таблицы

Col1	Col2	Col2	Col3
1	6	87837	787
2	7	78	5415
3	545	778	7507
4	545	18744	7560
5	88	788	6344

```
1  procedure RUN(packages, hashes)
2      queue[svace.parallel_max]
3      for item ∈ zip(packages, hashes)
4          ps = create(item)
5          if !queue.full()
6              queue.put(ps)
7          else
8              first = queue.get()
9              first.wait()
10         end if
11     end for
12 end procedure
```

Листинг 5 — Привер псевдокода на алгоритмическом языке

ПРИЛОЖЕНИЕ Б

Ещё один пример приложения

Пример приложения