



Министерство науки и высшего образования Российской Федерации  
Калужский филиал  
федерального государственного бюджетного  
образовательного учреждения высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК «Информатика и управление»

КАФЕДРА ИУК4 «Программное обеспечение ЭВМ, информационные технологии»

## ПРОИЗВОДСТВЕННАЯ ПРАКТИКА

### «Научно-исследовательская работа»

Студент гр. ИУК4-11М \_\_\_\_\_ ( Сафронов Н.С. )  
(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_ ( Гагарин Ю.Е. )  
(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_  
30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_  
30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_  
(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_ ( Белов Ю.С. )  
(подпись) (Ф.И.О.)

\_\_\_\_\_ ( Гагарин Ю.Е. )  
(подпись) (Ф.И.О.)

\_\_\_\_\_ ( Корнюшин Ю.П. )  
(подпись) (Ф.И.О.)

Калуга, 2024

УТВЕРЖДАЮ  
Заведующий кафедрой **ИУК4**  
\_\_\_\_\_(Гагарин Ю.Е.)  
« 05 » сентября 2024 г.

## **ЗАДАНИЕ** **на ПРОИЗВОДСТВЕННУЮ ПРАКТИКУ, НАУЧНО-** **ИССЛЕДОВАТЕЛЬСКУЮ РАБОТУ (НИР)**

За время выполнения НИР студенту необходимо:

- |  |  |
|--|--|
| <b>1.</b> определить объект и предмет исследования, используя информационные ресурсы, включая научно-технические литературные источники; изучить патентную документацию, составить аналитический обзор; выявить наличие и доступность ресурсов, необходимых для проведения теоретических и экспериментальных исследований; определить цели и задачи исследования;<br>сформулировать тему исследования; | <b>2.</b> выполнить анализ существующих результатов фундаментальных и поисковых исследований; выбрать и обосновать направления исследований и способов решения задач; разработать общую методику проведения исследований; выбрать последовательность и процедуры проведения исследований и обработки их результатов; |
|--|--|

*в том числе:*

- *Проанализировать существующие подходы и модели генерации программного кода на основе естественного языка.*
- *Изучить методы и алгоритмы, используемые для решения задач генерации программного кода, и оценить их эффективность.*
- *Оценить метрики и датасеты для обучения моделей генерации кода, а также выявить основные направления для дальнейших исследований.*

- 3.** Подготовить отчет и защитить результаты НИР.

Дата выдачи задания « 05 » сентября 2024 г.

Руководитель НИР \_\_\_\_\_ Гагарин Ю.Е.

Задание получил студент гр. ИУК4-11М \_\_\_\_\_ Сафронов Н.С.

## Оглавление

Введение .....	4
Работа с информационными ресурсами по теме исследования .....	5
Разработка плана исследований .....	8
Проведение исследований в соответствии с разработанным планом .....	10
Заключение .....	15
Список литературы .....	16

## **Введение**

Цель: сформировать навыки применения технологии взаимодействия с информационными ресурсами, в том числе в режиме удаленного доступа; навыки выполнения обзоров научно-технических литературных источников. Сформировать навыки составлять и реализовывать планы проведения исследований в выполняемых проектах; навыки применения современных методов исследований, разработки методики и организации проведения экспериментов. Сформировать навыки генерации различных вариантов решений; навыки планирования, проведения, анализа и интерпретации результатов теоретических и экспериментальных исследований.

Задачи: определить объект и предмет исследования, используя информационные ресурсы, включая научно-технические литературные источники; изучить патентную документацию, составить аналитический обзор; выявить наличие и доступность ресурсов, необходимых для проведения теоретических и экспериментальных исследований; определить цели и задачи исследования; сформулировать тему исследования. Выполнить анализ существующих результатов фундаментальных и поисковых исследований; выбрать и обосновать направления исследований и способов решения задач; разработать общую методику проведения исследований; выбрать последовательность и процедуры проведения исследований и обработки их результатов. Разработать рабочие гипотезы; обосновать принятые допущения; построить модели объекта исследований, провести теоретические исследования; определить необходимость проведения экспериментов для подтверждения положений теоретических исследований или для получения конкретных значений параметров, необходимых для проведения расчетов; разработать методики экспериментальных исследований, подготовить модели (макеты, экспериментальные образцы), а также подобрать испытательное оборудование.

## **Работа с информационными ресурсами по теме исследования**

Информация по теме собиралась на следующих ресурсах:

**Dehaerne, E., Dey, B., Halder, S., De Gendt, S., & Meert, W. (2022). Code Generation Using Machine Learning: A Systematic Review. IEEE Access.**

В статье представлен систематический обзор существующих методов генерации кода с использованием машинного обучения. Основное внимание уделено трем типам задач: генерация кода из естественного языка (description-to-code), создание документации из кода (code-to-description) и модификация существующего кода (code-to-code). Авторы анализируют различные модели машинного обучения, включая RNN, LSTM, transformers и CNN, а также подходы к токенизации и методы оценки качества синтезированного кода.

Статья представляет собой всесторонний обзор современных подходов к генерации кода с применением машинного обучения и подчеркивает важность выбора правильных моделей и методов токенизации. Работы в этой области продолжают развиваться, и предложенные направления исследований помогут улучшить эффективность и точность генерации кода. Статья будет полезна для исследователей и разработчиков, занимающихся автоматическим программированием и машинным обучением.

**Chen, M., et al. (2021). Evaluating Large Language Models Trained on Code.**

В статье рассматриваются возможности языковой модели Codex, специализированной на генерации программного кода, разработанной на основе GPT-3 и обученной на коде из открытых репозиториях GitHub. Основное внимание уделено задаче синтеза функций Python по текстовым описаниям (docstrings) и оценке функциональной корректности кода с использованием набора тестов HumanEval.

Модель Codex достигает существенного улучшения в точности синтеза кода по сравнению с другими моделями (например, GPT-3 и GPT-J). Также исследуются такие методы, как многократная генерация выборок для повышения качества решений. Среди ограничений модели выделены сложности с длинными цепочками операций и их привязкой к переменным.

Статья включает обсуждение этических, экономических и экологических последствий применения подобных технологий, а также возможных рисков, связанных с безопасностью и злоупотреблением. Работа представляет интерес для разработчиков, инженеров в области машинного обучения и специалистов по генерации кода.

**Gu, A., & Dao, T. (2023). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In Advances in Neural Information Processing Systems.**

В статье "Mamba: Linear-Time Sequence Modeling with Selective State Spaces" авторы Альберт Гу и Три Дао представляют новую архитектуру модели Mamba, которая обеспечивает линейное масштабирование по длине последовательности и высокую производительность на различных типах данных.

Модели на основе трансформеров страдают от квадратичной сложности вычислений на длинных последовательностях.

Mamba успешно решает задачи, требующие контекстно-зависимого рассуждения, и может экстраполировать решения на последовательности длиной более 1 миллиона токенов. Mamba превосходит трансформеры по качеству предсказаний и скорости генерации текста, показывая лучшие результаты на задачах обработки естественного языка, включая общие тесты на здравый смысл.

Представленная архитектура Mamba является первой моделью, которая линейно масштабируется по длине последовательности и достигает качества трансформеров на различных типах данных.

**Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. In Advances in Neural Information Processing Systems (NIPS).**

В статье представлена новая архитектура нейронных сетей для решения задач, требующих работы с последовательностями данных, таких как машинный перевод. Авторы предлагают использование многослойной архитектуры LSTM для кодирования и декодирования последовательностей. Ключевыми инновациями являются использование LSTM для преобразования входной последовательности в фиксированное представление и применение реверсирования порядка слов для улучшения результатов.

Работа открывает новые возможности для решения сложных задач с переменной длиной входных и выходных последовательностей и является важным вкладом в области обработки естественного языка.

**Ghosh, D., Zhu, P. H., & Bayley, I. (2024). Benchmarks and Metrics for Evaluations of Code Generation: A Critical Review. arXiv preprint arXiv:2400.08768.**

В статье проведен критический обзор существующих подходов к оценке моделей генерации программного кода, с фокусом на метрики и бенчмарки. Авторы анализируют различные типы задач программирования, а также большие языковые модели (LLMs) для программирования, такие как GPT-NEO, GPT-J и Codex. Рассматриваются проблемы с текущими метриками качества, такими как синтаксическая близость (BLEU, ROUGE-L) и функциональная корректность. Также подчеркивается важность создания разнообразных и объективных бенчмарков для оценки моделей, а также разработки новых метрик. В статье предлагаются направления для будущих исследований, включая улучшение тестовых наборов и использование метаданных для сценарных оценок.

## Разработка плана исследований

Был разработан подробный план исследований на тему генерации программного кода из естественного языка. План состоит из семи основных пунктов:

1. Анализ актуальности и значимости проблемы генерации программного кода на основе естественного языка. В этом пункте проводится исследование актуальности применения машинного обучения для генерации кода из естественного языка, а также обзор существующих тенденций и потребностей в области автоматизации разработки программного обеспечения.

2. Обзор моделей Sequence-to-Sequence с механизмом внимания. Рассматриваются основные принципы работы моделей типа Sequence-to-Sequence (seq2seq), включая их механизмы внимания, и то, как эти модели решают задачи, связанные с обработкой последовательностей, включая генерацию программного кода.

3. Обзор генеративных предварительно обученных трансформеров (GPT). В этом разделе изучаются архитектуры GPT (Generative Pretrained Transformer), которые применяются для генерации текстовых последовательностей, включая программный код. Рассматриваются особенности и преимущества моделей GPT в контексте генерации кода.

4. Изучение архитектуры Mamba для обработки длинных последовательностей данных. Здесь рассматривается новая архитектура Mamba, ориентированная на линейное масштабирование обработки длинных последовательностей и её применение для генерации кода.

5. Анализ датасетов для обучения моделей генерации кода. Описание используемых датасетов для тренировки моделей генерации кода, включая открытые репозитории и специфичные наборы данных, содержащие примеры кода и описание функций.

6. Метрики для оценки качества сгенерированного кода. Рассматриваются различные метрики для оценки качества генерируемого



кода, включая BLEU, ROUGE и другие, а также способы анализа функциональной правильности и тестирования сгенерированного кода.

7. Подведение итогов проведенного исследования.

## **Проведение исследований в соответствии с разработанным планом**

В первой главе работы была проведена всесторонняя обзорная работа по современным подходам к генерации программного кода с использованием методов машинного обучения. В результате анализа существующих методов было выявлено, что подходы на основе моделей Sequence-to-Sequence, таких как seq2seq с механизмом внимания, а также генеративные модели, такие как GPT, играют ключевую роль в этой области.

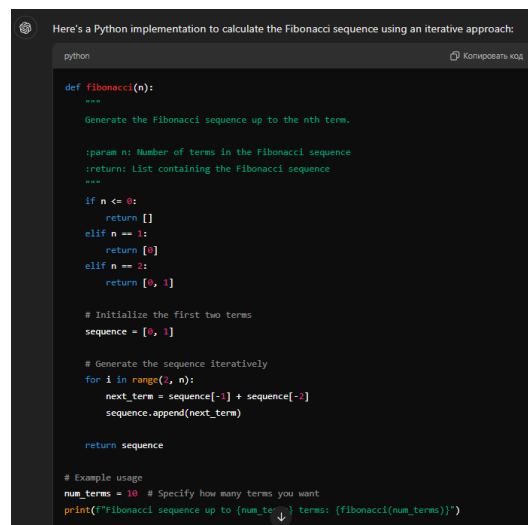
Модели Sequence-to-Sequence с механизмом внимания стали основой для многих успешных систем генерации кода. Например, работы Sutskever et al. (2014) показали, что применение этих моделей позволяет эффективно работать с длинными зависимостями в данных, что особенно важно для синтеза программного кода из текстовых описаний. Механизм внимания позволяет модели учитывать контекст всего входного текста, что значительно улучшает качество генерации. Это может быть экстраполировано на задачи генерации программного кода, где важно сохранять смысловую целостность и структуру.

Одним из наиболее значимых достижений в области генерации текста и кода является развитие моделей генеративного предварительно обученного трансформера (GPT). Эти модели, начиная с GPT-1 и до более новых версий, таких как GPT-3 и GPT-4, показали отличные результаты в различных задачах обработки естественного языка, включая генерацию текстовых последовательностей и кода. В отличие от традиционных моделей, GPT использует архитектуру трансформера с механизмом внимания, что позволяет эффективно работать с большими объемами данных и учитывать контекст на всех уровнях. Модели GPT обучаются на огромных наборах данных, включая тексты, коды и другие виды информации, что позволяет им генерировать осмысленные и грамматически правильные тексты, а также решать сложные задачи, такие как генерация программного кода.

Примером успешного применения GPT для генерации кода является модель Codex, разработанная на основе GPT-3. В исследовании Chen et al. (2021) была рассмотрена модель Codex, которая продемонстрировала

значительное улучшение в синтезе Python-кода по текстовым описаниям, по сравнению с более ранними моделями. Codex способна генерировать программный код, который соответствует ожиданиям пользователя и выполняет необходимые задачи. Модель была обучена на огромном количестве репозиторий с открытым исходным кодом, включая GitHub, что позволило ей разобраться в синтаксисе и логике различных языков программирования.

Одним из ключевых преимуществ GPT-4 и Codex является способность генерировать код с учетом контекста, включая поддержку различных языков программирования и стилей кодирования. Эти модели также могут быть использованы для автозаполнения, исправления ошибок, создания документации и перевода кода между языками программирования. Например, GPT-4 способен преобразовывать описание задачи на естественном языке в программный код, что значительно ускоряет процесс разработки и может быть использовано для обучения начинающих программистов.



```
python
Here's a Python implementation to calculate the Fibonacci sequence using an iterative approach:

def fibonacci(n):
    """
    Generate the Fibonacci sequence up to the nth term.

    :param n: Number of terms in the Fibonacci sequence
    :return: List containing the Fibonacci sequence
    """
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]

    # Initialize the first two terms
    sequence = [0, 1]

    # Generate the sequence iteratively
    for i in range(2, n):
        next_term = sequence[-1] + sequence[-2]
        sequence.append(next_term)

    return sequence

# Example usage
num_terms = 10 # Specify how many terms you want
print(f"Fibonacci sequence up to {num_terms} terms: {fibonacci(num_terms)}")
```

Рисунок 1 – Пример генерации кода ChatGPT, используя модель GPT-4

Тем не менее, несмотря на успехи GPT в генерации кода, существуют определенные ограничения. Во-первых, модели GPT, как и другие трансформеры, имеют сложности при работе с длинными последовательностями, особенно когда код требует обработки длинных зависимостей и сложных операций. Например, при генерации кода для

сложных алгоритмов или работы с большими объемами данных, модели GPT могут столкнуться с проблемами по обеспечению точности и структурной целостности. Во-вторых, модели GPT требуют большого количества вычислительных ресурсов для обучения и функционирования, что делает их менее доступными для широкого круга пользователей.

Другим важным направлением является использование модели Mamba, которая была представлена в исследовании Gu & Dao (2023). Модель Mamba решает проблему квадратичной сложности обработки длинных последовательностей в традиционных трансформерах, предлагая линейное масштабирование по длине последовательности. Это решение значительно повышает скорость генерации и точность предсказаний на длинных текстах, что может быть особенно полезным для задач генерации кода, где часто необходимо обрабатывать большие объемы информации. В экспериментах Mamba показала лучшие результаты по сравнению с традиционными трансформерами, что открывает новые возможности для улучшения качества генерации кода. Этот подход также справляется с контекстно-зависимым рассуждением и способен обрабатывать текстовые последовательности, которые могут достигать миллиона токенов, что делает её весьма перспективной для применения в задачах генерации сложных программных решений.

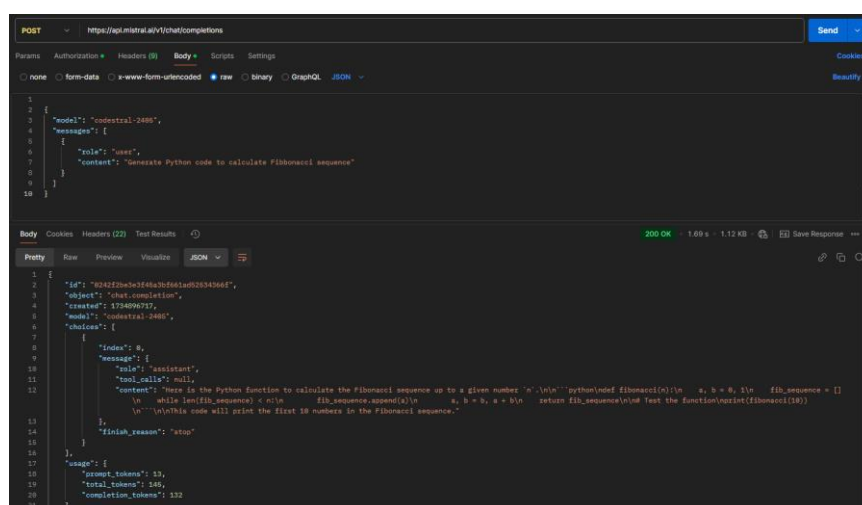


Рисунок 2 – Пример работы Codestral, используя модель Mamba, посредством API

Кроме того, в обзоре рассматривались различные датасеты, используемые для обучения моделей генерации кода. Одним из наиболее популярных является датасет HumanEval, который используется для оценки функциональной корректности сгенерированного кода. Этот и другие датасеты, такие как CodeXGLUE, включают разнообразные примеры кода, что позволяет моделям обучаться на реальных задачах и обеспечивать высокую точность при генерации кода. Chen et al. (2021) активно используют такие датасеты для оценки своих моделей, и результаты показывают, что использование качественных датасетов значительно улучшает производительность моделей.

Метрики оценки качества генерируемого кода также были важной частью исследования. В работах, таких как Dehaerne et al. (2022), широко применяются такие метрики, как BLEU, ROUGE, CIDEr и METEOR, которые традиционно используются для оценки качества сгенерированных текстов. Однако авторы отмечают, что для оценки качества программного кода эти метрики не всегда подходят, поскольку они не учитывают функциональную правильность кода. Вместо этого, для оценки программного кода важны такие метрики, как функциональная корректность, тестируемость и способность к выполнению задач. В связи с этим необходимо развивать новые метрики, которые будут учитывать особенности программирования и специфические требования к качеству кода.

Результаты исследований, представленных в первой главе, показывают, что генерация программного кода с использованием моделей машинного обучения, таких как seq2seq и GPT, достигла значительного прогресса. Однако, несмотря на достижения, существует ряд проблем, которые остаются актуальными. Во-первых, необходимо продолжать работу над улучшением функциональной корректности генерируемого кода, а также над увеличением точности моделей при обработке длинных цепочек операций. Во-вторых, необходимо совершенствовать метрики оценки качества кода, чтобы они более точно отражали функциональную и структурную целостность

сгенерированного программного продукта. Тем не менее, исследования, проводимые в этой области, открывают перспективы для дальнейшего улучшения и внедрения моделей генерации кода, что обещает значительные улучшения в области автоматического программирования.

Таким образом, в рамках проведенных исследований были рассмотрены ключевые подходы и модели, используемые для генерации программного кода, включая Sequence-to-Sequence модели, трансформеры, такие как GPT, и новые архитектуры, такие как Mamba. Результаты работы других авторов дают основу для дальнейших разработок и доработок в этой области, а также определяют перспективные направления для будущих исследований.

## Заключение

В ходе исследования были рассмотрены различные подходы к генерации программного кода с использованием современных моделей машинного обучения, таких как seq2seq, GPT, Mamba и другие. Особое внимание было уделено архитектурам, которые эффективно решают задачи синтеза кода, включая задачи генерации программного кода по текстовым описаниям.

Модели seq2seq, основанные на рекуррентных нейронных сетях, показали свою эффективность в обработке последовательностей данных, включая задачи перевода кода и генерации функций. GPT и аналогичные модели, такие как Codex, продемонстрировали значительные успехи в генерации кода, обучаясь на больших объемах данных, включая открытые репозитории и примеры программ. Эти модели, благодаря своей универсальности и способности работать с контекстом, значительно улучшили точность синтеза кода.

Архитектура Mamba, предлагающая линейное масштабирование и улучшенную производительность на длинных последовательностях, представила собой перспективное направление для дальнейших разработок в этой области. Она решает проблемы, с которыми сталкиваются трансформеры, и может быть применена для обработки более сложных задач с большими объемами данных.

Однако, несмотря на значительные достижения, существующие модели остаются уязвимыми к определенным ограничениям, таким как трудности с длинными зависимостями в коде и с функциональной корректностью. Необходимо продолжать развивать методы токенизации, улучшать метрики оценки качества синтезированного кода и разрабатывать более эффективные бенчмарки для точной оценки этих моделей.

Исследование показало значительный потенциал использования машинного обучения для генерации программного кода, что открывает перспективы для повышения автоматизации разработки программного обеспечения.

## Список литературы

1. Аверченков, В.И. Основы научного творчества [Электронный ресурс]: учеб. пособие/ В.И. Аверченков, Ю.А. Малахов. — Брянск: Брянский государственный технический университет, 2012. — 156 с.— Режим доступа: <http://www.iprbookshop.ru/7004>.
2. Рыжков, И.Б. Основы научных исследований и изобретательства [Электронный ресурс]: учебное пособие / И.Б. Рыжков. — 3-е изд., стер. — Санкт-Петербург : Лань, 2019. — 224 с.— URL: <https://e.lanbook.com/book/116011>
3. Астанина, С.Ю. Научно-исследовательская работа студентов (современные требования, проблемы и их решения) [Электронный ресурс]: монография/ С.Ю. Астанина, Н.В. Шестак, Е.В. Чмыхова. — М.: Современная гуманитарная академия, 2012.— 156 с.— Режим доступа: <http://www.iprbookshop.ru/16934>.
4. Комлацкий, В. И. Планирование и организация научных исследований [Электронный ресурс] : учебное пособие / В. И. Комлацкий, С. В. Логинов, Г. В. Комлацкий. — Ростов-на-Дону : Феникс, 2014. — 205 с. —URL: <http://www.iprbookshop.ru/58980.html>
5. Краснов, С.В. Методические указания по выполнению выпускной квалификационной работы (магистерской диссертации) для обучающихся по направлению подготовки 09.04.03 «Прикладная информатика» (уровень магистратуры) [Электронный ресурс]: учебно-методическое пособие / С.В. Краснов, Е.А. Матвеева, А.Р. Диязитдинова.— Самара: Поволжский государственный университет телекоммуникаций и информатики, 2015. — 23 с. —Режим доступа: <http://www.iprbookshop.ru/71853.html>
6. Мокий, М.С. Методология научных исследований [Текст]: учебник / М.С. Мокий, А.Л. Никифоров, В.С. Мокий. - М.: Юрайт, 2015. - 255 с.



7. Губарев, В.В. Квалификационные исследовательские работы [Электронный ресурс]: учеб. пособие/ В.В. Губарев, О.В. Казанская. — Новосибирск: Новосибирский государственный технический университет, 2014. — 80 с.— Режим доступа: <http://www.iprbookshop.ru/47691>.
8. Половинкин, А.И. Основы инженерного творчества [Электронный ресурс]: учебное пособие / А.И. Половинкин. — 7-е изд., стер. — Санкт-Петербург : Лань, 2019. — 364 с.— URL: <https://e.lanbook.com/book/123469>
9. Новиков, Ю. Н. Подготовка и защита бакалаврской работы, магистерской диссертации, дипломного проекта : учебное пособие для вузов / Ю. Н. Новиков. — 5-е изд. испр. и доп. — Санкт-Петербург : Лань, 2021. — 36 с. — ISBN 978-5-8114-4727-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/174283>
10. Рекомендации по написанию и оформлению курсовой работы, выпускной квалификационной работы и магистерской диссертации [Электронный ресурс]: учебно-методическое пособие/ Е.В. Зудина [и др.]. — Волгоград: Волгоградский государственный социально-педагогический университет, 2016. — 57 с.— Режим доступа: <http://www.iprbookshop.ru/57785>.