«Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И.Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Направление	09.03.04 - Программная инженерия
Профиль	Разработка программно-информационных систем
Факультет	КТИ
Кафедра	МО ЭВМ

К защите допустить

Зав. кафедрой А.А. Лисс

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Тема: Разработка модели искусственного интеллекта для прогнозирования поведения программ в ОС

Студент		_____ *подпись*	П.А. Бодунов
Руководитель	*(Уч. степень, уч. звание)*	_____ *подпись*	А.А. Лисс
Консультанты	*(Уч. степень, уч. звание)*	_____ *подпись*	А.В. Гаврилов
	(Уч. степень, уч. звание)	_____ *подпись*	И.И. Иванов
	(Уч. степень, уч. звание)	_____ *подпись*	И.И. Иванов

Санкт-Петербург

2024

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Утверждаю
Зав. кафедрой МО ЭВМ
_____ А.А. Лисс
«___»_____20___ г.

Студент Бодунов П.А. Группа 0303

Тема работы: Разработка модели искусственного интеллекта для прогнозирования поведения программ в ОС

Место выполнения ВКР: СПБГЭТУ «ЛЭТИ», кафедра МО ЭВМ

Исходные данные (технические требования):

Необходимо реализовать трассировку списка процессов, чтобы получать список последовательных событий (системных вызовов). Данный список будет базой для обучения модели искусственного интеллекта. Результатом работы должна быть программа, делающая предсказание следующего системного вызова для конкретной программы.

Содержание ВКР:

Кратко перечисляются основные разделы ВКР

Перечень отчетных материалов: пояснительная записка, иллюстративный материал, иные отчетные материалы

Дополнительные разделы: безопасность жизнедеятельности

Дата выдачи задания «___»_____20___ г.

Дата представления ВКР к защите «___»_____20___ г.

Студент _____ П.А. Бодунов

	_____	
Руководитель *(Уч. степень, уч. звание)*	_____	А.А. Лисс
Консультант *(Уч. степень, уч. звание)*	_____	А.В. Гаврилов

КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой аббревиатура названия кафедры

_____ И.И. Иванов

«___»_____20___ г.

Студент Иванов И.И. Группа 0000

Тема работы: Наименование темы

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	00.00 – 00.00
2	Наименование раздела	00.00 – 00.00
3	Наименование раздела	00.00 – 00.00
4	Наименование раздела	00.00 – 00.00
5	Оформление пояснительной записки	00.00 – 00.00
6	Оформление иллюстративного материала	00.00 – 00.00

Студент _____ И.И. Иванов

Руководитель _____ И.И. Иванов
(Уч. степень, уч. звание)

Консучёльтант _____ И.И. Иванов
(Уч. степень, уч. звание)

РЕФЕРАТ

Пояснительная записка 65 стр., 19 рис., 16 табл., 24 ист.

НЕЙРОННЫЕ СЕТИ, ТРАССИРОВКА, СИСТЕМНЫЕ ВЫЗОВЫ, ПРЕДСКАЗАНИЕ СИСТЕМНЫХ ВЫЗОВОВ.

Объектом исследования является модель прогнозирования системных вызовов.

Предметом исследования является точность и время предсказания системных вызовов.

Цель работы: разработка эффективной модели искусственного интеллекта для прогнозирования действий программ в реальном времени с целью улучшения производительности операционных систем.

~~Знание поведения программ полезно при разработке более совершенных методов планирования процессов в ОС. В данной выпускной квалификационной работе изучаются подходы к предсказанию поведения программ с целью выявления наиболее эффективного метода для прогнозирования системных вызовов. Проведено исследование различных аналогов для модели прогнозирования. Были рассмотрены предиктор последнего значения, предиктор на основе таблиц, унифицированный метод, предиктор на основе дерева решений, а также предсказание на основе Sequence-to-Sequence модели. По результатам анализа работы каждого аналога с точки зрения их точности и производительности, было принято решение использовать предиктор на основе дерева решений в качестве наиболее точной модели прогнозирования. Для выбранного метода прогнозирования был создан набор данных с помощью следующих утилит трассировки: strace и bpftrace. Полученный набор данных был разделен в соотношении 80% для обучения и 20% для оценки результатов обучения нейронной сети.~~

ABSTRACT

Перевод аннотации на английский язык.

СОДЕРЖАНИЕ

Для составления Содержания удобно использовать автоматически собираемое Оглавление — если вы корректно использовали соответствующие стили заголовков (согласно данному шаблону), то в Содержании автоматически войдут нужные разделы с номерами страниц (разделы Задание, Календарный план, Реферат, Abstract в Содержании не указываются).

Для заголовков первого уровня (названия глав) используйте стиль «Заголовок 2», для подразделов — «Заголовок 3», для подразделов следующих уровней — «Заголовок 4».

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ЯП – язык программирования;

IDE – Integrated Development Environment;

Токенизация – это процесс разделения текста на токены;

Токен – это единица обработки текста для анализа и обработки языка;

Математическая лингвистика – это область науки, которая изучает методы математики и статистики для анализа текстов, генерации и понимания языка, распознавания речи, и других задач, связанных с лингвистикой.

Natural language processing (NLP) - область математической лингвистики и искусственного интеллекта, которая занимается обработкой и анализом человеческого языка.

Word embedding - это процесс преобразования текстовой информации в числовой формат (числовые векторы в многомерном пространстве), который помогает компьютеру более эффективно понимать и обрабатывать естественный язык.

ВВЕДЕНИЕ (заголовок второго уровня)

Основной текст разделов, параметры — отступ от начала каждого абзаца 1,25 см, 14 шрифт TimesNewRoman, полуторный интервал, выравнивание по ширине, интервал перед и после абзаца нулевой. Эти настройки уже включены в стиль «Основной текст» данного документа.

Во Введении необходимо последовательно (и в разных абзацах) описать Актуальность, Цель работы, Задачи работы, Объект исследования, Предмет исследования, Практическую ценность работы, перечислить публикации по теме работы (при наличии).

Обращаем ваше внимание, что сторонние инструменты для редактирования текстовых файлов (например, Google Docs) могут игнорировать встроенные шрифты, заменяя их собственными (меняя характеристики/названия/пр.), что может привести к некорректному оформлению и ошибках при автоматической проверке.

Современные операционные системы сталкиваются с растущим объемом данных и сложностью работы приложений, что может привести к снижению производительности и увеличению времени отклика системы. Предсказание поведения программ позволяет оптимизировать производительность операционной системы и приложений, путем уменьшения загрузки CPU, улучшения использования памяти и уменьшения времени ожидания при выполнении системных вызовов [1], [2], [3].

Цель работы заключается в создании эффективной модели искусственного интеллекта, которая может прогнозировать действия программ в реальном времени, способствуя улучшению производительности операционных систем.

Целью работы является повышение точности модели искусственного интеллекта, которая может прогнозировать действия программ в реальном

времени, способствуя улучшению производительности операционных систем.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- Реализовать трассировку списка процессов, чтобы получать список последовательных событий (системных вызовов).
- Обработать список системных вызовов и разделить их на обучающую, валидационную и тестовую выборки, тем самым подготовив данные для обучения модели ИИ.
- Подобрать модель искусственного интеллекта.
- Обучить модель на обучающем наборе данных.
- Оценить качество модели ИИ на тестовой выборке.

Модель прогнозирования системных вызовов в ОС Linux является объектом исследования.

Предметом исследования является точность и время предсказания системных вызовов.

1 Обзор предметной области

1.1 Роль системных вызовов в ОС

Операционная система Linux одна из наиболее широко используемых операционных систем [7], относится к семейству Unix-подобных операционных систем с открытым исходным кодом, основанных на ядре Linux. Система состоит из следующих основных компонентов:

1. Ядро Linux - основа операционной системы, управляющая работой аппаратного обеспечения, обеспечивающая основные функции операционной системы и отвечающая за выполнение прикладных программ.

2. Командная оболочка (shell) - интерфейс командной строки, с помощью которого пользователи взаимодействуют с операционной системой.

3. Системные утилиты — это набор базовых программ, который обеспечивает выполнение основных функций операционной системы (управление файлами, сетевое взаимодействие, администрирование системы и другое).

4. Графическое окружение - набор программ и библиотек, обеспечивающих графический интерфейс пользователя (GUI) для удобства работы с операционной системой.

5. Драйверы устройств - программное обеспечение, обеспечивающее взаимодействие операционной системы с аппаратным обеспечением компьютера.

Для обеспечения стабильной и эффективной работы операционной системы Linux эти компоненты работают вместе.

Операционная система обладает двумя режимами работы: пользовательский режим и режим ядра.

Пользовательский режим - это режим работы, в котором выполняются пользовательские процессы или пользовательские программы. В этом режиме процессы имеют ограниченный доступ к ресурсам компьютера и не имеют прямого доступа к аппаратному обеспечению. Пользовательский режим обеспечивает изоляцию процессов друг от друга и обеспечивает безопасность системы.

Режим ядра - это режим работы, в котором выполняется только ядро операционной системы. В данном режиме ядро операционной системы имеет полный контроль над ресурсами компьютера и осуществляет их управление. Ядро операционной системы управляет обработкой прерываний, переключением контекста процессов, управлением памятью и другими низкоуровневыми операциями.

Пользовательский режим взаимодействует с режимом ядра для обеспечения работы операционной системы. Ядро операционной системы предоставляет интерфейс для взаимодействия между пользовательскими программами и аппаратным обеспечением компьютера, обеспечивает безопасность и стабильность работы системы.

Пространство пользователя может взаимодействовать с пространством ядра посредством системных вызовов.

Системный вызов – это функция операционной системы, которая позволяет приложению или пользовательскому процессу взаимодействовать с ядром операционной системы и получать доступ к различным ресурсам и функциям системы. Системные вызовы обеспечивают интерфейс для работы с аппаратным обеспечением компьютера через операционную систему.

Состояние процесса в конкретный момент времени называется «контекстом».

Переключение контекста(context switch) – это процесс, при котором операционная система переключает выполнение процессора с выполнения одного процесса на другой. При переключение контекста выполнение текущего процесса останавливается и его контекст сохраняется, чтобы когда процесс будет возобновлен, он продолжил работу с того же момента, на котором был прерван.

Стоимость контекстного переключения (context switch) зависит от множества факторов и считается затратной операцией [8], так как требует сохранения и загрузки состояния процесса, включая значения регистров, указатели стека, таблицы страниц памяти и другие данные. Уменьшение ча-

стоты переключения контекста является ключевым аспектом повышения производительности системы.

Предсказание следующего системного вызова способствует уменьшению частоты переключения контекста и повышению производительности программы. Когда предсказание делается точно, операционная система может заранее загружать данные в память и подготавливать необходимые ресурсы, что позволяет уменьшить время ожидания выполнения операций и ускорить работу программы.

1.2 Существующие модели предсказания

1.2.1 Предиктор последнего значения

Предиктор последнего значения [1] предсказывает, что значение текущего экземпляра инструкции будет таким же, как значение, полученное предыдущим экземпляром. Этот предиктор применяет функцию идентификации к ранее наблюдаемому значению и использует его в качестве следующего прогноза. Этот метод довольно простой и полезен, когда в данных наблюдается устойчивая тенденция и незначительная изменчивость.

1.2.2 Предиктор на основе таблиц

Программы при выполнении демонстрируют различную степень периодичности, что указывает на то, что модели поведения повторяются с течением времени. Предикторы на основе таблиц разработаны на использовании этой повторяемости. Они используют историю прошлого поведения в качестве индекса в таблице прогнозирования. Предсказание, сохраненное в таблице является значением, которое соответствовало истории поведения в прошлом.

При выполнении новой рабочей нагрузки предикторы проходят начальную фазу обучения, чтобы заполнить таблицу прогнозов историческими шаблонами. Для получения прогнозов на этапе обучения может использоваться схема Предиктор последнего значения. Предикторы адаптируются к измене-

ниям в изученном поведении, обновляя записи в таблице, но необходимо соблюдать осторожность, чтобы избежать обновления таблицы ложными значениями из-за шумовых пиков. В дополнение к прогнозируемому значению, в каждой записи таблицы хранится последнее фактическое значение, которое наблюдалось в истории. После получения нового фактического значения, таблица обновляется путем установки на наиболее частое из трех значений: прогнозируемого и последнего значения, сохраненные в таблице, и нового фактического значения.

1.2.3 Унифицированный метод прогнозирования (UPM)

UPM - это универсальный метод прогнозирования, который не зависит от системы и компьютерных показателей. В основе метода лежит параметрическая модель, основанная на статистической базе.

Применение UPM состоит из нескольких этапов.

Первым шагом для применения UPM является выбор параметрической модели (например, авторегрессионная, марковская). Наиболее часто применяется авторегрессионная модель, которая требуют очень мало информации о статистике входных данных и имеет простую реализацию.

Авторегрессионная модель зависит только от предыдущих выходных данных системы и позволяет находить коэффициенты линейных предикторов, которые предсказывают текущее значение на основе прошлых образцов, причем предполагается, что параметры модели постоянны на протяжении всего периода анализа.

Вторым шагом является определение целевой функции (например, функция среднеквадратичной ошибки, функция накопленной квадратичной ошибки, вероятность). Целевая функция, как правило, является функцией параметров модели и прошлых наблюдаемых выборок.

Наиболее часто используемой целевой функцией является функция среднеквадратичной ошибки, которая имеет множество преимуществ, в том числе следующее:

- Минимизация целевой функции приводит к линейной системе уравнений, которую легко решить без значительных затрат. Любой другой выбор целевой функции может привести к нелинейной системе уравнений, которая может быть решена с помощью численной оптимизации. Решение нелинейной системы уравнений может быть дорогостоящим без гарантий сходимости.
- Целевая функция предпочитает множество небольших ошибок нескольким большим, которые линейно складываются в одно и то же значение, что обычно желательно для минимизации влияния превышений в прогнозировании.
- Функция среднеквадратичной ошибки монотонно растет со стандартным отклонением, поэтому минимизация квадрата ошибки также сводит к минимуму другие статистические показатели, которые обычно используются.

Третьим шагом является оценка параметров модели путем минимизации выбранной целевой функции. Она может быть минимизирована либо аналитически, либо с помощью численной оптимизации. Форма решения определяется выбранной целевой функцией и типом модели.

1.2.4 Предиктор на основе дерева решений

Предиктор на основе дерева решений используются методы машинного обучения (дерево решений) построенное при помощи алгоритма C4.5. Для построения датасета используется набор определенных статических и динамических атрибутов процессов во время их выполнения. В работе [] были проведены исследования по выявлению наиболее важных статических и динамических характеристик процессов, которые могут наилучшим образом помочь в прогнозировании времени загрузки процессора и минимизировать время выполнения процесса. Результатом работы стал набор атрибутов для построения датасета, представленный в табл. 1, который был получен с помощью команд readelf и size.

Таблица 1 – Набор атрибутов для построения датасета для предиктора на основе дерева решений

Атрибут	Его ранг	Обозначение
InputSize	1	Значение размера входных данных, зависящее от типа программы
ProgramName	2	Название программы и номинальный атрибут
BSS	3	Информация о размере неинициализированных данных
RoData	4	Размер данных (байт), доступный только для чтения, который обычно используется для создания недоступного для записи сегмента в образе процесса
Text	5	Текст или исполняемые инструкции
InputType	6	Тип входного номинального значения

Обучение осуществляется путем анализа статических и динамических атрибутов процессов, представленных в таблице.

1.2.5 Предсказание на основе Sequence-to-Sequence модели

В своей архитектуре Sequence-to-Sequence модель [4] использует рекурентные нейронные сети (RNN) для обработки последовательностей входных и выходных данных. В этой работе для улучшения производительности базовых классификаторов обнаружения вторжений авторы объединили предсказанную последовательность с вызванной последовательностью, чтобы сформировать расширенные входные данные. Как показано на рис. 1, предсказанная последовательность предоставляет классификаторам дополнительную информацию для улучшения возможности определения характера последовательности системных вызовов.

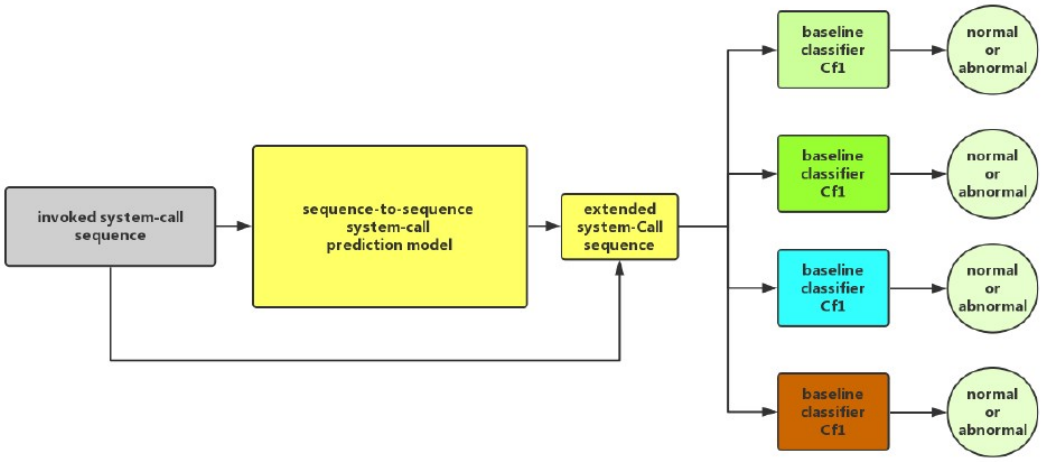

Рисунок 1 – Схема работы Sequence-to-Sequence модели

Авторы рассматривают трассировки системных вызовов, создаваемые во время работы программы, как набор языковых предложений, которые процесс использует для взаимодействия с операционной системой. Для генерации семантически обоснованных последовательностей системных вызовов внедряется структура sequence-to-sequence языковой модели RNN с «механизмом внимания» (attention) (кодер выполняет операцию прямого распространения). Механизм attention позволяет подчеркнуть влияние некоторых определенных системных вызовов, которые лучше отражают намерения программы, на конкретное декодирование системных вызовов. Таким образом данный метод использует информацию о входных данных, которые наиболее важны при декодировании каждого системного вызова.

Архитектура модели sequence-to-sequence с attention показана на рис. 4.

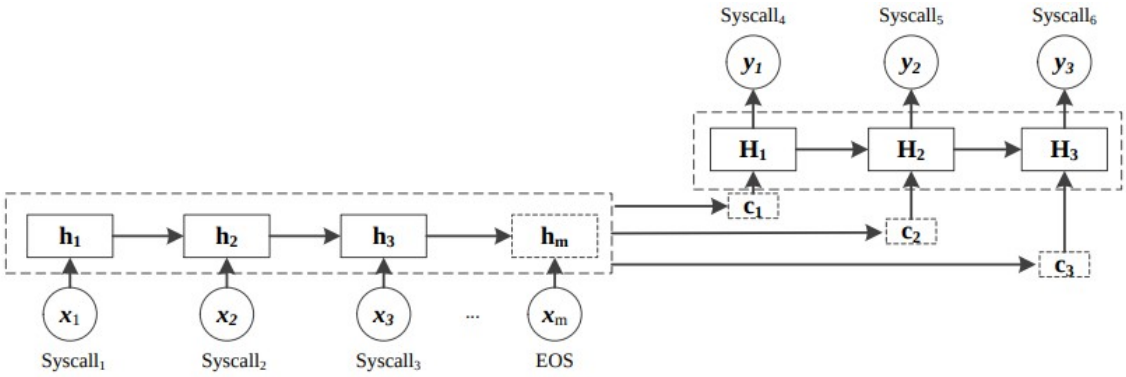

Рисунок 2 – Схема работы Sequence-to-Sequence модели с attention

Где h_i – i-ое скрытое состояние. $i \in (1,2,\ldots,m)$, а c_i обозначает i-й информационный вектор контекста, который вычисляется с помощью алгоритма attention.

Предложенная прогностическая модель предсказывает, какие системные вызовы будут выполнены впоследствии, что позволит наблюдать состояние системы и прогнозировать атаки.

1.3 Критерии сравнения аналогов

1.3.1 Точность прогнозирования

Точность прогнозирования системных вызовов зависит от нескольких факторов, включая используемые алгоритмы прогнозирования, объем доступных данных и качество обучающей выборки. Чтобы оценить точность прогнозирования, данные разделяют на обучающий и тестовый наборы. Одним из распространенных методов оценки точности прогнозирования является процент правильно предсказанных системных вызовов относительно общего числа вызовов.

1.3.2 Лицензирование ОС

Лицензирование операционной системы — это процесс получения разрешения на использование операционной системы. Существуют два основных типа лицензий: проприетарные и открытые. Проприетарные лицензии обычно требуют оплаты и предоставляют пользователю ограниченные права на использование программного обеспечения. Открытые лицензии, такие как лицензии GNU General Public License (GPL), предоставляют пользователю более широкие права, включая право на изменение и распространение программного обеспечения.

Сравнительная характеристика аналогов представлена в табл. 2:

Таблица 2 – Сравнение аналогов

Критерий/Аналог	Точность	Лицензирование ОС
Предиктор последнего значения	88%	Коммерческое

Предиктор на основе таблиц	92%	Коммерческое
Унифицированный метод прогнозирования	94%	Коммерческое
Предиктор на основе дерева решений	94%	Коммерческое/Свободное
Предсказание на основе Sequence-to-Sequence модели	93%	Коммерческое/Свободное

1.3.2 Лицензирование ОС

Предиктор последнего значения обладает наименьшей точностью среди остальных методов. Предиктор на основе таблиц, унифицированный метод, предиктор на основе дерева решений, предсказание на основе Sequence-to-Sequence модели обладают практически одинаковой точность. Предиктор последнего значения, предиктор на основе таблиц, унифицированный метод прогнозирования используются в архитектуре IBM Power и ОС AIX, которая является коммерческой проприетарной операционной системой, разработанной компанией IBM для серверов, имеет высокую производительность, надежность и поддержку различных аппаратных компонентов. Предиктор на основе дерева решений, предсказание на основе Sequence-to-Sequence модели используются в архитектуре x86 и ОС Linux, которая является свободно распространяемой операционной системой с открытым исходным кодом и может быть использована на различных типах компьютеров или серверов, имеет большое сообщество разработчиков и пользователей, что приводит к быстрому внедрению новых технологий и исправлению ошибок.

2 Формулировка требований к решению и постановка задачи

2.1 Постановка задачи

Необходимо собрать набор данных системных вызовов, разработать модель искусственного интеллекта для прогнозирования следующего системного вызова по пяти вызовам и оценить время предсказания.

2.2 Требования к решению

На основе сравнения аналогов решение должно удовлетворять следующим условиям:

1. Повышенной точностью прогнозирования.
2. Обучение модели на датасете системных вызовов, содержащих информацию о контексте, включая указание на используемое приложение и выполняемое действие.
3. Обеспечение более точной работы обученной нейросети за счет большого количества тестов.

2.3 Обоснование требований к решению

1. Модель ИИ должна обладать повышенной точностью прогнозирования, не меньше, чем на существующих решениях, чтобы успешно предсказывать следующие системные вызовы.
2. Обучение модели должно происходить на данных системных вызовов, содержащих информацию о контексте, включая указание на используемое приложение и выполняемое действие для более точного предсказания.
3. Модель должна быть обучена на большом количестве тестов для охвата каких-нибудь уникальных тестов для улучшения точности предсказания системных вызовов.

2.4 Обоснование выбора метода решения

Для выполнения условия повышенной точности выбраны нейронные сети. Нейронные сети могут обнаруживать сложные нелинейные зависимости между переменными, что может быть полезно при анализе системных вызовов, которые могут зависеть друг от друга и от различных внешних факторов, автоматически извлекать важные признаки из данных, что может помочь в повышении точности предсказаний системных вызовов. NLP м

3 Проектирование модели ИИ

3.1 Подготовка данных

Для более точного прогнозирования системных вызовов данные должны обладать достаточно большим контекстом. Для системных вызовов это: название процесса, который вызывает системный вызов, название системного вызова, его аргументы и его результат выполнения.

Название последующего системного вызова необходимо предсказывать по нескольким системным вызовам, в данной работе предсказание осуществляется по пяти системным вызовам.

3.1.2 Токенизация

Разделение текста на токены является одним из первых и важных этапов при обработке текстов и для многих методов анализа естественного языка.

В процессе токенизации слов текст разделяется на отдельные слова, символы или фразы в зависимости от конкретной задачи. Токенизация слов может включать в себя удаление знаков препинания, приведение всех слов к нижнему регистру, удаление стоп-слов.

Последовательность из системных вызовов объединяется в предложение, которое следует обработать для нейронной сети, так же необходимо обработать результат нейронной сети (название системного вызова).

В данной задаче токенами будут считаться:

- Название процесса
- Название системного вызова
- Полное перечисление аргументов
- Результат системного вызова

Пример разбиения данных на токены представлен на рис. __.:

(картинка)

3.1.3 Word embedding

Для успешного обучения нейронной сети необходимо преобразовать токены в вектора чисел, это можно осуществить при помощи embedding. Вектора получившиеся после word embedding обладают следующими свойствами:

- Семантическая близость. Слова с похожим значением имеют близкие векторы в пространстве, что позволяет измерять семантическую близость между словами.
- Арифметические операции над векторами слов позволяют находить аналогии.

Данные свойства могут быть полезны из-за существования системных вызовов, которые выполняют задачу с похожим смыслом, но при этом имеют разные названия.

Пример word embedding представлен на рис. __:

(картинка)

3.2 Архитектура и принцип работы Нейронной сети

Рисунок 3 – Пример рисунка

Каждая таблица должна иметь ссылку в тексте (таблица 1).

Таблица 1 – Пример таблицы

Показатель	Единица измерения	Значение
Время работы	сек.	159
Нагрузка на ЦПУ	процент	95
Стоимость работ	тыс. руб.	100

4 Реализация

4.1 Сбор датасета

4.2 Реализация нейронной сети

4.3 Используемые технологии

5 БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

ЗАКЛЮЧЕНИЕ (заголовок второго уровня)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ (заголовок второго уровня)

1. Duesterwald E., Cascaval C., Dwarkadas S. Characterizing and predicting program behavior and its variability //2003 12th International Conference on Parallel Architectures and Compilation Techniques. – IEEE, 2003. – С. 220-231.
2. Sarikaya R., Buyuktosunoglu A. A unified prediction method for predicting program behavior //IEEE Transactions on Computers. – 2009. – Т. 59. – №. 2. – С. 272-282.
3. Negi A., Kumar P. K. Applying machine learning techniques to improve linux process scheduling //TENCON 2005-2005 IEEE Region 10 Conference. – IEEE, 2005. – С. 1-6.
4. Lv S. H. et al. Intrusion prediction with system-call sequence-to-sequence model //IEEE Access. – 2018. – Т. 6. – С. 71413-71421.
5. Xu C. et al. Cache contention and application performance prediction for multi-core systems //2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS). – IEEE, 2010. – С. 76-86.
6. Thaker N., Shukla A. Python as multi paradigm programming language // International Journal of Computer Applications. – 2020. – Т. 177. – №. 31. – С. 38-42.
7. https://gs.statcounter.com/os-market-share/desktop/worldwide
8. https://dl.acm.org/doi/abs/10.1145/1281700.1281704
9.

ПРИЛОЖЕНИЕ А (заголовок второго уровня)