Конструктор блоков обработки естественного языка и применение его в задаче структурирования логов в информационной безопасности

В.В. Чаругин, В.В. Чаругин, А.Н. Чесалин, Н.Н. Ушкова

Аннотация **– В работе исследуется проблема обработки естественного языка в информационной безопасности.**

В статье предлагается конструктор блоков обработки естественного языка, описывается его концепция, архитектура и принцип работы. Рассматривается решение задачи структурирования логов в информационной безопасности, используя разработанный конструктор. Формируется единый и стандартизированный формат для записи событий.

Проводится анализ моделей естественного языка (BERT, ALBERT, DistilBERT, XLNet, GPT-2) для задачи структурирования логов. Производится оценка качества алгоритмов с использованием метрик: Accuracy и F1-Score.

Результаты задачи структурирования логов могут быть использованы аналитиками и разработчиками в информационной безопасности, а также применены в расширении функциональности SIEM-системы.

Ключевые слова **– алгоритмы обработки естественного языка, трансформеры, BERT, ALBERT, DistilBERT, XLNet, GPT-2, анализ данных, структурирование данных, конструктор блоков обработки, машинное обучение, информационная безопасность.**

I. Введение

В современном мире объем текстовой информации стремительно растет. Обработка и анализ такого огромного объема требует использования различных технологий обработки естественного языка, которые позволяли бы информационным системам понимать и интерпретировать текст на уровне, сравнимом с человеческим. С увеличением объема текстовой информации растет и количество различных задач, которые необходимо эффективно решать. Таким образом, появляется необходимость в создании гибкой и масштабируемой системы, способной выполнять обработку естественного языка с помощью функциональных блоков (пайплайнов) обработки данных. Данный подход позволит не только повысить эффективность и точность решений задач естественного языка, но и обеспечить возможность быстрой адаптации под новые требования.

В данной работе предлагается реализация конструктора блоков обработки естественного языка, описывается его архитектура и алгоритм работы, и демонстрируется практическое применение его на конкретной задаче – структурирование логов в информационной безопасности.

II. Задачи обработки естественного языка

Существует множество задач обработки естественного языка [1], к примеру: морфологический анализ, синтаксический анализ, семантический анализ, извлечение именованных сущностей (NER), классификация, обобщение, генерация текста, машинный перевод, распознавание текста и речи и пр. Для решения задач обработки естественного языка разработано и используется огромное количество моделей, методов и алгоритмов, при этом одни и те же модели, методы и алгоритмы используются как основа для других методов, либо применяются в совокупности. Практически всегда решение прикладных задач обработки естественно языка – это сложная комбинация моделей, методов и алгоритмов.

Для примера, рассмотрим некоторые прикладные задачи искусственного интеллекта в информационной безопасности и комбинации алгоритмов обработки естественного языка, применяемых для их решения:

1. Задача обнаружения фишинговых атак с использованием искусственного интеллекта

Этапы решения задачи:

1. Извлечение URL-адресов.
Используется алгоритм NER для извлечения URL-адресов из писем.
2. Анализ URL-адресов.
Применяется алгоритм анализа URL-адресов для проверки их подлинности и связи с фишинговыми сайтами.
3. Классификация электронных писем.
Используется алгоритм классификации для определения писем, которые могут быть фишинговыми.
4. Обобщение результатов.
Обобщение результатов анализа для выявления типичных сценариев фишинговых атак, разработка стратегий защиты и рекомендаций по улучшению безопасности.

Алгоритм решения задачи обнаружения фишинговых атак представлен на рисунке 1.

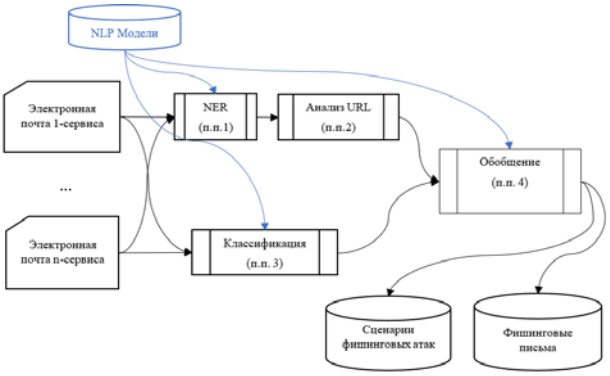

Рисунок 1 – Алгоритм обработки естественного языка в решении задачи обнаружения фишинговых атак

2. Задача структурирования логов в информационной безопасности из разных источников с использованием искусственного интеллекта.

Этапы решения задачи:

1. Определение источника лога.

Классификация лога для определения источника лога.

2. Определение основной информации.

Извлечение именованных сущностей для определения из логов основной информации: даты и время логирования, уровень важности, идентификатор процесса, пользователь, модуль ПО, событие.

3. Определение дополнительной информации.

Извлечение именованных сущностей для определения из логов следующей дополнительной информации: ip-адрес источника и назначения, порт, доменное имя.

4. Представление информации в удобной форме для анализа.

Создание записей для логов по определенному заданному шаблону

Алгоритм решения задачи структурирования логов в информационной безопасности представлен на рисунке 2.

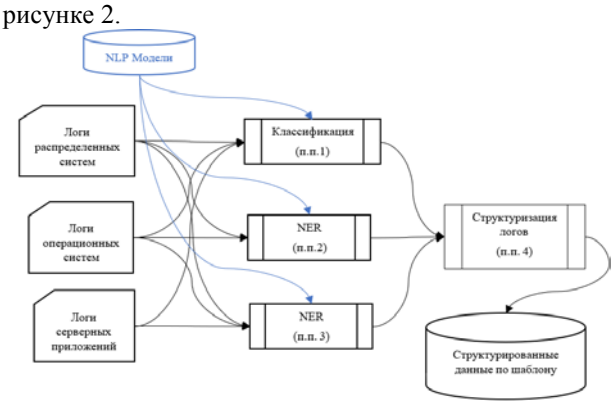

Рисунок 2 – Алгоритм обработки естественного языка в решении задачи структурирования логов информационной безопасности

Можно отметить, что решение прикладных задач обработки естественного языка представляет сложный конвейер обработки данных, в котором одновременно (параллельно, последовательно, повторно) может быть использовано большое количество наборов данных, алгоритмов обработки, моделей, и для эффективного решения подобных задач требуется автоматизация процесса обработки данных. В связи с этим предлагается конструктор блоков обработки естественного языка, реализованный в виде специального программного обеспечения, который позволяет:

– применять различные модели, методы и алгоритмы обработки естественного языка;

– проектировать сложные процессы обработки естественного языка, используя визуальное программирование (Low-code) и унифицированный интерфейс взаимодействия;

– оценивать качество моделей;

– встраивать процесс обработки естественного языка в реальные системы.

Анализ программного обеспечения в области обработки естественного языка показал, что на данный момент существуют системы визуального программирования, но только для задач машинного обучения и визуализации данных (к примеру, Orange, RapidMiner, Tableu, Loginom и пр.). Для решения сложных задач обработки естественного языка с использованием современных моделей глубокого обучения и больших лингвистических моделей таких систем нет.

III. Конструктор блоков обработки естественного языка

Конструктор блоков обработки естественного языка (Конструктор) представляет собой платформу визуального программирования (программное обеспечение) для решения различных комплексных задач. В нем используются современные алгоритмы обработки языка и открытые языковые модели.

Конструктор состоит из следующих компонентов и модулей:

– web-интерфейс;

– модуль обработки;

– база данных;

– модуль алгоритмов обработки естественного языка;

– модуль оценки точности.

Архитектура Конструктора представлена на рисунке 3.

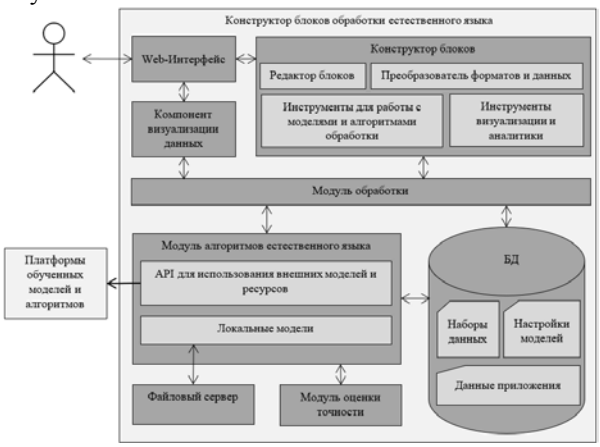

Рисунок 3 – Архитектура конструктора блоков обработки естественного языка

Конструктор включает в себя web-интерфейс, который предоставляет доступ к компоненту визуализации данных и конструктору блоков.

Компонент визуализации данных представляет собой компонент, который отвечает за преобразование и отображение записей наборов данных в удобной для человека форме.

Конструктор блоков – это инструмент, который предоставляет возможность создавать, настраивать и комбинировать различные блоки для обработки данных и анализа.

Конструктор включает в себя базу данных (БД), которая содержит наборы данных, модели, а также данные необходимые для работы приложения.

Набор данных состоит из записей заданной предметной области (к примеру, в [2] представлен процесс формирования набора данных для обнаружения сетевых аномалий), которые представляются в структурированной форме – «Запись NLP». «Запись NLP» представляет собой модель, состоящую из поля текста, полей и сущностей, предназначенных для работы с алгоритмами обработки естественного языка. Данный компонент позволяет упростить работу с текстовой информацией, делая ее более структурированной и подходящей для дальнейшего анализа.

Представление наборов данных представлено на рисунке 4.

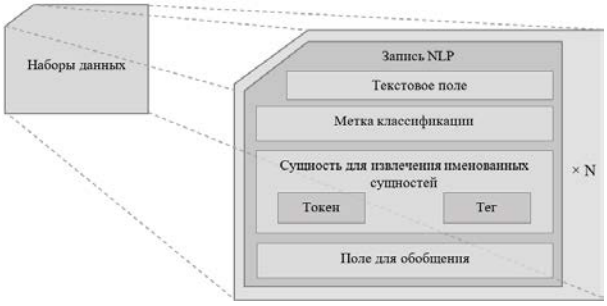

Рисунок 4 – Наборы данных

Конструктор включает в себя модуль обработки, который представляет собой блок, предназначенный для работы с наборами данных и их записями («Запись NLP»).

Модуль обработки выполняет следующие функции:

1) заполнение полей и сущностей в «Записях NLP», предназначенных для работы с алгоритмами обработки естественного языка;

2) копирование наборов данных;

3) очищение полей и сущностей в «Записях NLP», предназначенных для работы с алгоритмами обработки естественного языка, в наборах данных;

4) создание нового набора данных путем взятия результатов из конкретного поля «Записи NLP»;

5) удаление лишних записей в наборе данных – записей, не содержащих ни одного заполненного поля или сущностей, предназначенных для работы с алгоритмами обработки естественного языка;

6) автоматическое проставление тегов в «Записях NLP» на основе паттерна;

7) создание нового набора данных на основе шаблона и д. р.

Модуль алгоритмов обработки естественного языка в Конструкторе включает в себя различные модели машинного обучения. Модуль позволяет не только создавать и обучать собственные модели машинного обучения для обработки естественного языка, но и использовать и дообучать предобученные модели, взятые из различных открытых платформ (Например, HuggingFace). Данный модуль предназначен для решения задач обработки естественного языка, таких как классификация текстов, извлечение именованных сущностей и обобщение текстовых данных.

Для решения вышеперечисленных задач широкое применение получили генеративные модели, основанные на трансформерах [3]. За счет своей архитектуры они могут распараллеливать вычисления, что позволяет ускорить процесс обучения. Среди всех моделей-трансформеров выделяются следующие модели:

– BERT – генеративная модель, которая обучается с использованием маскированного языкового моделирования, где слова в предложениях маскируются, и модель должна предсказать замаскированные слова на основе контекста. Ее особенностью является учет контекста с обеих сторон от слова в предложении при обучении [4].

– ALBERT – модель BERT, в которой используются подходы: факторизация матрицы внимания и деление эмбеддингов слов на две независимые матрицы. Благодаря данным подходам модель уменьшает количество используемых параметров, сохраняя при этом высокую точность в задачах обработки естественного языка [5].

– DistilBERT – модель BERT, которая обучается с использованием техник дистилляции. Дистилляция – это процесс передачи информации от большой модели с целью создания компактной и эффективной модели [6].

– XLNet – модель глубокого обучения, которая использует механизм внимания и идеи из Transformer-XL. Модель использует авторегрессионный подход для предсказания следующего токена на основе предыдущих, учитывая контекст из разных мест исходной последовательности [7].

– GPT-2 – генеративная модель, которая предсказывает следующее слово на основе контекста предыдущего. Модель состоит из нескольких слоев, которые позволяют учитывать контекстуальные зависимости между словами в предложении [8].

Генеративные алгоритмы, используемые в задачах информационной безопасности, представлены в работе [9].

Модуль оценки точности в Конструкторе содержит метрики для моделей из модуля алгоритмов естественного языка. Модуль включает следующие метрики:

– Метрика точность (Accuracy). Она используется для задач классификации текстов. Метрика вычисляется как отношение количества правильно классифицированных текстов к общему количеству текстов.

– Метрика F1-мера. Она используется для задач классификации текстов. Данная метрика учитывает ложноотрицательные и ложноположительные ошибки.

– Метрика seqeval. Она используется для задач извлечения именованных сущностей. Метрика оценивает, насколько хорошо модель может правильно распознавать и классифицировать последовательности токенов в тексте как части определенных сущностей. Она включает в себя метрики точность и F1-меру для последовательностей.

– Метрика rouge. Она используется для задач обобщения текстовых данных. Данная метрика сравнивает полученный обобщенный текст с исходным, вычисляя, насколько хорошо текст воспроизводит ключевые фразы, сегменты или слова из исходного [10].

IV. Алгоритм работы конструктора блоков обработки естественного языка

Перед использованием моделей в блоках Конструктора необходимо их обучить под конкретные задачи. Обученные модели будут находиться в модуле алгоритмов естественного языка Конструктора.

При поступлении информации в текстовом виде на Конструктор создаются специальные набор(ы) данных с «Записями NLP». Созданные набор(ы) данных передаются на модуль обработки.

Модуль обработки представляет собой компонент, отвечающий за модификацию одного или нескольких наборов данных с «Записями NLP» с использованием определенной модели, взятой из модуля алгоритмов естественного языка. С помощью модуля обработки создаются цепочки операций, в результате которых формируются новые наборы данных. Эти новые наборы данных могут впоследствии использоваться в качестве исходных для создания следующих наборов данных.

Работа конструктора обработки естественного языка и модуля обработки представлена в блок-схемах на рисунках 5 и 6.

Рисунок 5 – Блок-схема алгоритма работы конструктора обработки естественного языка

Рисунок 6 – Блок-схема алгоритма работы модуля обработки

V. Структурирование логов в информационной безопасности из разных источников с применением искусственного интеллекта, используя Конструктор

Задача структурирования логов в информационной безопасности из разных источников заключается в том, чтобы извлечь и структурировать информацию из логов, чтобы системы безопасности и аналитики ИБ могли эффективно анализировать данные, своевременно выявлять аномалии и реагировать на угрозы.

Решение задачи структурирования логов с помощью Конструктора позволяет создавать единый и стандартизированный формат для записи событий. Устраняется проблема с внедрением новых технологий и продуктов, которые используют свои системы логирования. Подготовка индивидуальных правил для анализа и обработки логов каждой системы становится не только неудобной, но и крайне затратной задачей. Автоматическое структурирование логов позволит упростить процесс управления логами, снизить затраты на поддержку и повысить эффективность обнаружения и реагирования на инциденты без необходимости создания специфических решений для каждой системы.

Структурирование логов способствует решению следующих задач:

1) Определение атак, направленных на сокрытия следов.

Конструктор позволяет определить единый формат для всех записей логов в системе, что облегчит анализ и мониторинг логов, так как все события будут иметь одинаковую структуру и содержать необходимую информацию. В результате, злоумышленники не смогут легко изменить содержимое логов или маскировать события под законные действия, так как это будет выделяться на фоне стандартной структуры логов.

2) Выявление некорректной работы систем.

Структурирование логов с использованием Конструктора позволяет гарантировать полноту и целостность записей. Обнаружение аналитиками ИБ пропуска записей или неполных данных в логах, может свидетельствовать о проблеме в системе, такой как сбой системы или атака.

Результаты, полученные в задаче структурирования логов, могут быть эффективно применены в централизованном мониторинге безопасности. Они могут использоваться в следующих системах:

1) SIEM-системы. SIEM-системы могут использовать структурированные логи для выявления попыток несанкционированного доступа, атак на сеть или нарушений политик безопасности.

2) IDS-системы. IDS-системы могут использовать структурированные логи для обнаружения сканирования портов, попыток SQL-инъекций или других типов атак.

Структурированные логи также облегчают создание отчетов и проведение аудита безопасности. Они предоставляют полную и целостную информацию о событиях в системах, что позволяет проводить анализ и оценку безопасности не только одной системы, но и всего комплекса систем.

VI. Анализ логов в информационной безопасности с помощью Конструктора

Рассматривается следующий пример: в системе информационной безопасности собираются логи следующих систем Apache, HDFS, Hadoop, Linux, Mac, OpenSSH, Proxifier, Spark, Windows, Zookeeper. Требуется провести структурирование логов (создать единый и стандартизированный формат для записи событий) и реализовать механизмы выявления некорректной работы (аномалий) как для рассмотренных систем, так и для подключаемых других систем.

Для решения данной задачи предлагается применить комбинация алгоритмов классификации текста и извлечения именованных сущностей.

Алгоритмы классификации используются для определения источника лога. Данная информация позволяет системам безопасности и аналитикам ИБ определить уровень доверия к логу, а также при расследовании инцидентов безопасности сузить круг подозрительных событий и активностей.

Алгоритм извлечения именованных сущностей используется для определения основной (даты и время логирования, уровень важности, идентификатор процесса, пользователь, модуль ПО, событие) и дополнительной (ip-адрес источника и назначения, порт, доменное имя) информации лога. Основная информация позволяет системе безопасности лучше понять контекст событий и связать их с потенциальными угрозами или нарушениями. Дополнительная информация позволяет выявить источник угроз (данные за пределом контура безопасности), а также цель (данные в пределах контура безопасности).

В данном примере рассматриваются логи из различных источников, которые собраны в [11]. В таблице 1 представлены примеры логов и источники.

Таблица 1 – Примеры логов и источники

Источник	Пример лога
Apache	[Sun Dec 04 07:45:45 2005] [error] [client 63.13.186.196] Directory index forbidden by rule: /var/www/html/
HDFS	081109 235322 3968 INFO dfs.DataNode$DataXceiver: Receiving block blk_-9103764593642564543 src: /10.251.43.115:46385 dest: /10.251.43.115:50010
Hadoop	2015-10-18 18:01:52,728 INFO [main] org.mortbay.log: Started HttpServer2$SelectChannelConnectorWithSafeStartup @0.0.0.0:62267
Linux	Jun 14 15:16:02 combo sshd(pam_unix)[19937]: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=218.188.2.4
Mac	Jul 3 15:42:59 calvisitor-10-105-160-237 kernel[0]: ARPT: 682634.998705: wl0: setup_keepalive: Remote IP: 17.249.28.93
OpenSSH	Dec 10 07:07:45 LabSZ sshd[24206]: Failed password for invalid user test9 from 52.80.34.196 port 36060 ssh2
Proxifier	[10.30 16:49:09] chrome.exe - proxy.cse.cuhk.edu.hk:5070 open through proxy proxy.cse.cuhk.edu.hk:5070 HTTPS
Spark	17/06/09 20:10:46 INFO rdd.HadoopRDD: Input split: hdfs://10.10.34.11:9000/pjhe/logs/2kSOSP.log:21876+7292
Windows	2016-09-29 02:04:35, Info CBS Session: 30546354_3312401639 initialized by client WindowsUpdateAgent.
Zookeeper	2015-07-29 19:04:12,394 - INFO [/10.10.34.11:3888:QuorumCnxManager$Listener@493] - Received connection request /10.10.34.11:45307

Логи из различных источников, представленные в таблице 1, разнообразны по своей структуре. Для анализа логов аналитикам ИБ и системам безопасности необходимы структурированные данные, содержащие информацию о системе и о событии, произошедшем в системе. Для структурирования логов используется шаблон в формате json, представленный в листинге 1 [12, 13].

Листинг 1 – Шаблон структуризации лога

```
{
        "Source": "<classification_label>",
        "Datetime": "<Datetime>",
        "Level": "<Level>",
        "Process": "<Process>",
        "User": "<User>",
        "Node": "<Node>",
        "Component": "<Component>",
        "Content": "<Content>",
        "Ip": "<Ip>",
        "Port": "<Port>",
        "Domain": "<Domain>"
}
```

В данном шаблоне определяется источник (Source) как результат работы алгоритма классификации лога. Также в шаблоне выделяются общие теги логов: 8 тегов основной информации (Datetime, Level, Process, User, Node, Component, Content, Other) и 3 тега дополнительной информации (Ip, Port, Domain). Теги основной и дополнительной информации логов рассматриваются в разных наборах данных («Главный» и «Дополнительный»), так как информация пересекается в рамках одного тега, что может привести к некорректной обработке и интерпретации данных при использовании алгоритма распознавания именованных сущностей.

Примеры логов из наборов данных представлены на рисунках 7 и 8.

Рисунок 7 – Пример лога из «Главного» набора данных

Рисунок 8 – Пример лога из «Дополнительного» набора данных

Сформированные логи разбиваются на тренировочные, валидные и тестовые данные. Тренировочные данные содержат по 200 записей каждого источника, кроме Spark (суммарно 1800 записей). Валидные и тестовые содержат по 40 записей каждого источника (суммарно 400 записей). Apache, Linux, Spark обозначены как неизвестные источники, для них вводится метка классификации Unknown. Последнее условие добавляется для того, чтобы помочь моделям классификации естественного языка обобщить свои знания, позволяя им лучше обрабатывать новые данные из различных источников, с которыми они не были знакомы во время обучения. Сформированные наборы данных логов выложены на портале github: https://github.com/val-ugs/UNLP/tree/prod-v1/backend/dataset_examples/log_dataset/datasets.

Для формирования шаблона используется схема, представленная на рисунке 9.

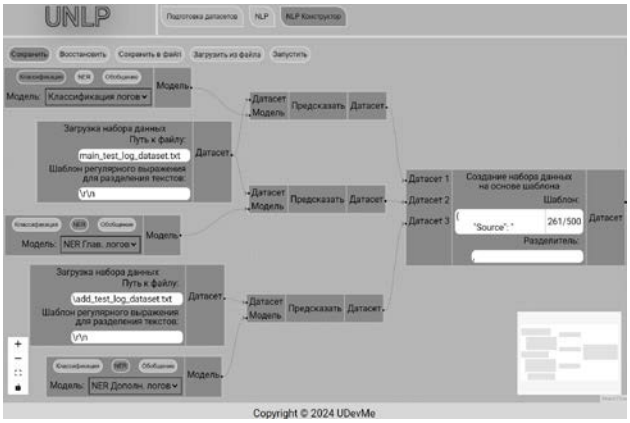

Рисунок 9 – Схема формирования шаблона

В схеме на рисунке 9 будут использоваться модели, которые продемонстрируют лучшие результаты для решения задач классификации и извлечения именованных сущностей. Для определения таких моделей проводится сравнение современных алгоритмов обработки естественного языка на основе тестовых данных.

В таблице 2 представлены результаты моделей обработки естественного языка для классификации. В таблице 3 и 4 представлены результаты алгоритмов обработки естественного языка для распознавания именованных сущностей для «Главного» и «Дополнительного» наборов данных.

Таблица 2 – Результаты работы моделей обработки естественного языка для классификации

Модель	Метрика		Время, затраченное на обучение	Время, затраченное на прогнозирование
	Accuracy	F1 score		
BERT	0.9	0.93	00:10:10:84	00:00:01:53
ALBERT	0.943	0.958	00:09:52:04	00:00:01:75
DistilBERT	0.951	0.962	00:04:52:04	00:00:00:86
XLNet	0.9	0.921	00:24:50:12	00:00:02:37
GPT-2	0.905	0.932	00:11:24:82	00:00:01:69

Таблица 3 – Результаты работы моделей обработки естественного языка для распознавания именованных сущностей для «Главного» набора данных

Модель	Метрика		Время, затраченное на обучение	Время, затраченное на прогнозирование
	Accuracy	F1 score		
BERT	0.964	0.916	00:12:09:43	00:00:01:53
ALBERT	0.967	0.921	00:11:47:96	00:00:01:78
DistilBERT	0.965	0.918	00:04:45:30	00:00:00:95
XLNet	0.944	0.872	00:21:50:12	00:00:02:30
GPT-2	0.969	0.927	00:17:20:48	00:00:01:58

Таблица 4 – Результаты работы моделей обработки естественного языка для распознавания именованных сущностей для «Дополнительного» набора данных

Модель	Метрика		Время, затраченное на обучение	Время, затраченное на прогнозирование
	Accuracy	F1 score		
BERT	0.999	0.99	00:06:30:54	00:00:02:73
ALBERT	0.999	0.994	00:06:12:34	00:00:01:34
DistilBERT	0,999	0,993	00:02:43:14	00:00:01:14
XLNet	0.998	0.987	00:10:21:42	00:00:03:07
GPT-2	0.999	0.991	00:08:54:42	00:00:02:81

На основе таблиц 2, 3, 4 построены диаграммы 1, 2, 3.

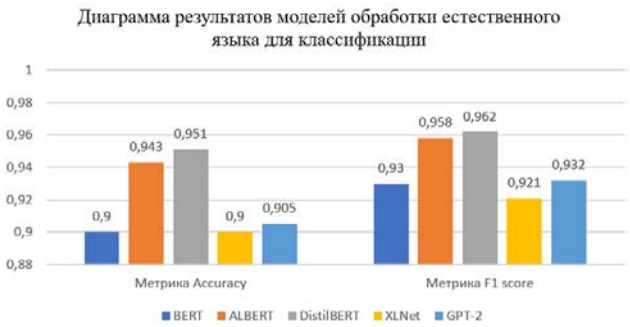

Диаграмма 1 – Диаграмма результатов моделей обработки естественного языка для классификации

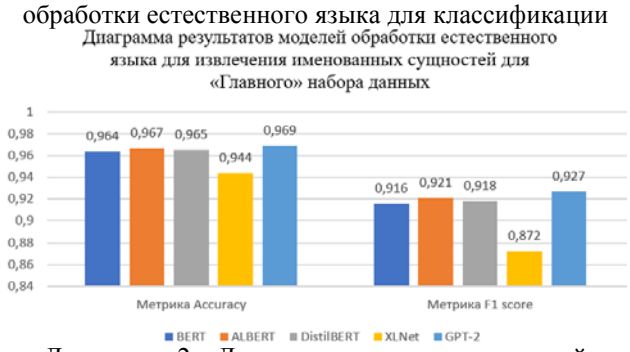

Диаграмма 2 – Диаграмма результатов моделей обработки естественного языка для извлечения именованных сущностей для «Главного» набора данных

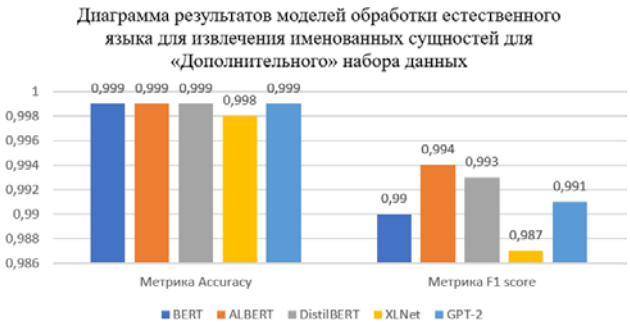

Диаграмма 3 – Диаграмма результатов моделей обработки естественного языка для извлечения именованных сущностей для «Дополнительного» набора данных

Результаты анализа показывают, что все рассмотренные модели обработки естественного языка имеют схожее качество в классификации и извлечении именованных сущностей. Лучшие результаты для классификации и для извлечения именованных сущностей получены при использовании DistilBERT. Он значительно превосходит остальные модели по скорости обучения, сохраняя при этом высокий уровень точности.

Результат работы с использованием модели DistilBERT в алгоритмах классификации и извлечении именованных сущностей для лога “Dec 10 09:15:04 LabSZ sshd[24555]: Failed password for root from 187.141.143.180 port 36419 ssh2\n” из источника OpenSSH представлены в листинге 2.

Листинг 2 – Результат работы конструктора блоков обработки естественного языка


```
{
            "Source": "OpenSSH",
            "DateTime": "1900-10-18T18:01:52.728",
            "Level": "",
            "Process": "24555",
            "User": "",
            "Node": "",
            "Component": "LabSZ",
            "Content": "Failed password for root from
187.141.143.180 port 36419 ssh2",
            "Ip": "187.141.143.180",
            "Port": "36419",
            "Domain": ""
}
```


VII. Заключение

В работе предложен конструктор блоков обработки естественного языка, рассмотрена его концепция, архитектура и принцип работы.

Представлен пример систематизации логов в информационной безопасности с помощью конструктора блоков обработки естественного языка. Применение конструктора блоков обработки естественного языка позволило автоматизировать процесс обработки данных используя визуальное программирование (Low-code) и унифицированный интерфейс взаимодействия.

Проведено тестирование моделей обработки естественного языка в задаче структурирования логов в информационной безопасности. Наилучшее качество показала модель DistilBERT.

Конструктор блоков обработки естественного языка может быть применен для решения сложных задач, состоящих из нескольких подзадач, позволяя настраивать каждую нейросетевую модель под конкретную подзадачу.

Результаты задачи структурирования логов могут быть использованы аналитиками и разработчиками в информационной безопасности, а также применены в расширении функциональности SIEM-системы.

Дальнейшее исследование в данном направлении лежит в развитии автоматизации процесса обработки естественного языка и в создании конструктора для мультимодальной обработки данных.

Библиография

[1] Christopher D. Manning and Hinrich Schütze. Foundations of Statistical Natural Language Processing. The MIT Press. ISBN 978-0-262-13360-9, 1999.

[2] Чаругин В.В., Чесалин А.Н. Анализ и формирование наборов данных сетевого трафика для обнаружения компьютерных атак // International Journal of Open Information Technologies ISSN: 2307-8162 vol. 11, no.6, 2023

[3] Vaswani A. et al. Attention is All you Need (англ.) // Advances in Neural Information Processing, 2017. — arXiv:1706.03762

[4] Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding// NAACL-HLT, 2019, pp 4171-4186.

[5] Zhenzhong Lan et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations// ICLR 2020.

[6] Victor Sanh et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter // NeurIPS 2019.

[7] Zhilin Yang et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding // NeurIPS, 2019.

[8] Alec Radford et al. Language Models are Unsupervised Multitask Learners. Preprint, 2019.

[9] Чаругин В.В., Чесалин А.Н. Применение генеративных алгоритмов в задачах информационной безопасности // «Фундаментальные, поисковые, прикладные исследования и инновационные проекты» Сборник трудов Национальной научно-практической конференции, 2023.

[10] Metrics. Available: https://huggingface.co/metrics (URL)

[11] Jieming Zhu et al. Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics. 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). 2023, pp 355-366.

[12] Structured Logging. Available: https://sematext.com/glossary/structured-logging/ (URL)

[13] Log Formatting in Production: 9 Best Practices. Available: https://betterstack.com/community/guides/logging/log-formatting/ (URL)

Статья получена: 20 мая 2024 года.

В.В. Чаругин – МИРЭА – Российский технологический университет (email: valentin.1999@mail.ru),
В.В. Чаругин – МИРЭА – Российский технологический университет (email: valera.2001@mail.ru),
А.Н. Чесалин – МИРЭА – Российский технологический университет (email: chesalin_an@mail.ru),
Н.Н. Ушкова - – МИРЭА – Российский технологический университет (email: ushkova.n.n@mail.ru),

Constructor of natural language processing blocks and its application in the problem of structuring logs in information security

V.V. Charugin, V.V. Charugin, A.N. Chesalin, N.N. Ushkova

***Abstract* – The paper examines the problem of natural language processing in information security.**

The paper proposes a constructor of natural language processing blocks, describes its concept, architecture and operating principle. The solution to the problem of structuring logs in information security using the developed constructor is considered A unified and standardized format for recording events is being formed.

The analysis of natural language models (BERT, ALBERT, DistilBERT, XLNet, GPT-2) for the task of structuring logs is carried out. The quality of algorithms is evaluated using the following metrics: Accuracy and F1-Score.

The results of the log structuring task can be used by analysts and developers in information security, and can also be used to expand the functionality of the SIEM-system.

***Keywords* – natural language processing algorithms, transformers, BERT, ALBERT, DistilBERT, XLNet, GPT-2, data analysis, data structuring, block processing constructor, machine learning, information security.**

REFERENCES

[1] Christopher D. Manning and Hinrich Schütze. Foundations of Statistical Natural Language Processing. The MIT Press. ISBN 978-0-262-13360-9, 1999.

[2] Charugin V.V., Chesalin A.N. Analysis and creation of network traffic datasets to detect computer attacks // International Journal of Open Information Technologies ISSN: 2307-8162 vol. 11, no.6, 2023

[3] Vaswani A. et al. Attention is All you Need (англ.) // Advances in Neural Information Processing, 2017. — arXiv:1706.03762

[4] Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding// NAACL-HLT, 2019, pp 4171-4186.

[5] Zhenzhong Lan et al. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations// ICLR 2020.

[6] Victor Sanh et al. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter // NeurIPS 2019.

[7] Zhilin Yang et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding // NeurIPS, 2019.

[8] Alec Radford et al. Language Models are Unsupervised Multitask Learners. Preprint, 2019.

[9] Charugin V.V., Chesalin A.N. Application of generative algorithms in information security problems // "Fundamental, exploratory, applied research and innovative projects" Collection of proceedings of the National Scientific and Practical Conference, 2023.

[10] Metrics. Available: https://huggingface.co/metrics (URL)

[11] Jieming Zhu et al. Loghub: A Large Collection of System Log Datasets for AI-driven Log Analytics. 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE). 2023, pp 355-366.

[12] Structured Logging. Available: https://sematext.com/glossary/structured-logging/ (URL)

[13] Log Formatting in Production: 9 Best Practices. Available: https://betterstack.com/community/guides/logging/log-formatting/ (URL)