Цель работы – разработка программного комплекса, позволяющего автоматизировать сбор и структурирование информации на естественном языке с использованием концепции машинного обучения.

В результате проделанной работы была проанализирована предметная область и современное состояние исследований в области агрегации данных и категоризации текстов, спроектирован пользовательский интерфейс, а также проведена апробация системы.

Эффективность разработанной системы заключается в автоматизации сбора структурированной информации с тематических интернет-ресурсов, обеспечения оптимальной организации процесса сбора данных, повышения удобства работы с информацией для пользователя.

Разработанный программный комплекс может быть применен для решения задачи автоматической категоризации данных из полученных документов на естественном языке.

# **ВВЕДЕНИЕ**

В настоящее время массивы информации, доступные человеку, многократно выросли благодаря развитию сети Интернет. Классификация/рубрикация информации (отнесение порции информации к одной или нескольким категориям из ограниченного множества) является традиционной задачей организации знаний и обмена информацией. В больших информационных коллекциях имеет смысл говорить только об автоматической рубрикации.

Поиск нужной информации среди большого объёма данных на множестве интернет-ресурсов становится все более сложной и затратной по времени задачей для человека. Информацию необходимо собирать, извлекать ее из множества неструктурированных источников близких между собой тематик, обрабатывать, структурировать, стремиться к повышению удобства и эффективности ее восприятия.

Решение этих задач возможно с применением обработки естественных языков и реализацией ее алгоритмов. Одна из наиболее важных задач обработки естественных языков, связанная с агрегацией данных и необходимая для ее реализации – это извлечение именованных сущностей.

***Цель работы -*** разработка программного комплекса, позволяющего автоматизировать сбор и структурирование информации на естественном языке с тематических интернет-ресурсов – классификатора данных.

Для достижения поставленной цели необходимо решить ***следующие задачи***:

1. Проанализировать современное состояние исследований в области агрегации данных и анализа естественных языков.
2. Провести анализ существующих научных и практических решений в выбранной области, изучить методы, принципы и технологии извлечения именованных сущностей, возможности их применения для агрегации данных.
3. Спроектировать компонентную реализацию программной системы для агрегации тематических данных.
4. Разработать программный комплекс.
5. Сделать вывод об эффективности созданной системы классификации данных.

***Решение поставленных задач определило структуру работы.***

В первой части проведено исследование ключевых аспектов применения автоматизированного анализа текстов, произведен обзор методов и алгоритмов извлечения информации из текстов на естественном языке. Во второй – представлена архитектура системы классификации текстов на естественном языке, её программная реализация.

В результате решением поставленных задач стал программные комплекс, позволяющий классифицировать тексты на естественном языке по различным тематикам.

***Объект исследования*** – классификация неструктурированных массивов информации

***Предмет исследования*** – методы и алгоритмы обработки естественных языков и классификации текстов.

***Методология и методы исследования.*** При выполнении работы использованы методы классификации текстов на естественном языке с использованием технологии машинного обучения, объектно-ориентированное программирование, системный анализ.

***Научная новизна и теоретическая значимость работы.*** Новизна данной работы заключается в следующем:

* Спроектирована новая компонентная реализация программной системы для классификации полученных данных, отличиемкоторой является наличие взаимодействующих масштабируемых модулей парсинга текста на естественном языке и работы с базой данных.
* Разработан программный комплекс, реализующий классификацию данных из текстов на естественном языке на основе алгоритма Байеса.
* Проведено тестирование программного комплекса на практических примерах.

***Практическая значимость.*** В ходе выполнения работы был создан программный комплекс – автоматический классификатор данных с применением алгоритмов анализа естественного языка, применимый для извлечения структурированной информации из текстов. Данный программный комплекс позволяет автоматически обрабатывать поступающие материалы по выбранным тематикам.

# **Анализ предметной области**

* 1. **Концепция машинного обучения**

Классификация или рубрикация информации: отнесение порции информации к одной или нескольким категориям из ограниченного множества, является традиционной задачей организации знаний и обмена информацией. В больших информационных коллекциях имеет смысл говорить только об автоматической рубрикации.

В настоящее время предложено много методов для решения данной задачи посредством автоматических процедур. Существующие методы можно разделить на два принципиально различных класса: методы машинного обучения и методы, основанные на знаниях, также иногда именуемые «инженерный подход».

При применении методов машинного обучения для построения классификатора используется коллекция документов, предварительно отобранная человеком. Алгоритм машинного обучения строит процедуру классификации документов на основе автоматического анализа заданного множества текстов.

При использовании методов, основанных на знаниях, правила отнесения документа к той или иной рубрике задаются экспертами на основе анализа рубрикатора и, возможно, части текстов, подлежащих рубрицированию.

Любые методы автоматической классификации текстов в той или иной форме используют знания о свойствах текста на естественном языке и знания об особенностях текстов, принадлежащих той или иной рубрике.

Принципиальная разница между двумя группами методов состоит в том, что методы машинного обучения используют математические методы для извлечения знаний из обучающей коллекции текстов, в то время как «экспертный подход» использует знания специалиста о свойствах текстов, принадлежащих рубрикам. Знания эксперта основываются, в первую очередь, на предыдущем опыте, в частности, на большой коллекции прочитанных ранее текстов, и во вторую очередь, на части текстов, подлежащих рубрицированию.

В исследованиях, посвященных применению методов машинного обучения для классификации текстов, применяются универсальные алгоритмы, которые применимы для широкого круга задач анализа и обработки информации. Для задачи классификации текстов эти методы работают с абстрактной векторной моделью документа и не учитывают особенностей задачи тематической классификации текстов и структуры рубрикатора. Тем не менее, во многих случаях методы машинного обучения дают весьма высокие результаты. Качество рубрикации для систем, основанных на машинном обучении, является довольно высоким для небольших рубрикаторов, и сильно падает с увеличением количества рубрик и усложнением структуры рубрикатора.

Во многих случаях, даже при наличии заранее отобранной коллекции документов, методы машинного обучения неприменимы и используется значительно более трудоемкий «экспертный», который обычно обеспечивает высокое качество рубрицирования и "прозрачность" алгоритма — результаты обработки легко.

* 1. **Концепция машинного обучения**

Машинное обучение ‒ это научное исследование алгоритмов и статистических моделей, которые компьютерные системы используют для эффективного выполнения конкретной задачи без использования явных инструкций, опираясь на шаблоны и выводы.

Абстрактный процесс машинного обучения начинается с подачи входных данных для алгоритма. Эти данные могут быть разными, начиная с более простых, такие как числовые или текстовые данные, заканчивая более сложными, например, видеофайл или изображение. Эти данные считываются электронно-вычислительной машиной и впоследствии обрабатываются различными математическими и статистическими алгоритмами. После обработки компьютер выводит эти же данные в виде информации, которые могут быть полезны для анализа, принятия решений, иллюстрации и так далее.

Машинное обучение лежит на стыке информатики, инженерии и статистики и часто появляется в других дисциплинах. Это инструмент, который может быть применен ко многим проблемам. Любая область, которая должна интерпретировать информацию и действовать на основе полученных данных, может извлечь выгоду из методов машинного обучения.

Машинное обучение является важным компонентом науки о данных, которая сейчас развивается стремительными темпами. С помощью статистических методов алгоритмы обучаются классифицировать данные, строить прогнозы и выделять важные моменты в ходе проектов по сбору и анализу данных.

Алгоритмы машинного обучения строят математическую модель на основе выборочных данных, известных как «обучающие данные», чтобы делать прогнозы или решения без явного программирования для выполнения задачи.

Задачи, которые способно решить машинное обучение, напрямую определяют выгоды для бизнеса и возможности решения социальных проблем государствами разных стран. К основным задачам относятся:

**Регрессия**. Предоставляет прогноз на основе выборки объектов с различными признаками. По итогам анализа данных на выходе получается число или числовой вектор. Например, таким образом работает кредитный скоринг — оценка кредитоспособности потенциального заёмщика.

**Классификация**. Выявляет категории объектов на основе имеющихся параметров. Продолжает традиции машинного зрения, поэтому часто можно встретить термин «распознавание образов»: например, идентификация разыскиваемых людей по фото или на основании словесного описания внешности.

**Кластеризация**. Разделяет данные на схожие категории по объединяющему признаку. Например, космические объекты кластеризируют по удаленности, размерам, типам и другим признакам.

**Идентификация**. Отделяет данные с заданными параметрами от остального массива данных. К примеру, участвует в постановке медицинского диагноза по набору симптомов.

**Прогнозирование**. Работает с объемами данных за определенный период и предсказывает на основе анализа их значение через заданный период времени. Примером может служить прогноз погоды.

Извлечение знаний. Исследует зависимости между рядом показателей одного и того же явления или события. Например, находит закономерности во взаимодействии биржевых показателей.

## **Машинное обучение для анализа текстовых данных**

Большое количество знаний и информации, хранится в текстовых документах. Чаще всего эти тексты доступны только в неструктурированном представлении, так как они создаются и интерпретируются людьми. Для того, чтобы воспользоваться преимуществами этого огромного количества скрытой информации, и включить ее в аналитические процессы, необходимо преобразовать информацию в структурированное представление. IE решает именно эту задачу. Система IE пытается идентифицировать четко определенные объекты и отношения в неструктурированных данных, особенно в текстовых документах.

Обработка естественного языка (Natural Language Processing) ‒ интеллектуальный анализ данных и методы машинного обучения используются вместе для автоматической классификации и обнаружения шаблонов из электронных документов. NLP решает большой набор задач, который можно разбить по уровням (в скобках). Среди этих задач, можно выделить следующие:

* Распознавание текста, речи, синтез речи (сигнал);
* Морфологический анализ, канонизация (слово);
* POS-тэгирование, распознавание именованных сущностей, выделение слов (словосочетание);
* Синтаксический разбор, токенизация предложений (предложение);
* Извлечение отношений, определение языка, анализ эмоциональной окраски (абзац);
* Аннотация документа, перевод, анализ тематики (документ);

NLP — это огромная область и, конечно, в ней применяются самые разные методы. Начиналось все с простейших алгоритмов, основанных на правилах и словарях, которые по заданным синтаксическим и семантическим шаблонам искали в текстах определенные конструкции или слова, на основе которых алгоритм и давал ответ. А сегодня в этой области применяются большие языковые модели, основанные на нейросетях.

Языковая модель — это такая модель, которая для фрагмента текста (а в NLP текст чаще всего рассматривают как последовательность слов) умеет оценивать вероятность встретить такую последовательность слов в языке. Как следствие языковая модель умеет находить наиболее вероятное продолжение для текста. Но для того, чтобы нейросеть могла генерировать продолжение текста, ее необходимо «обучить». Для этого модель «тренируют» на гигантских объемах текстов, написанных на естественном языке, и таким образом, основываясь на этих текстах, модель постепенно выучивает вероятности встретить одни слова рядом с другими, а значит и каким образом можно продолжить текст. То есть мы моделируем язык при помощи вероятности встретить одни слова рядом с другими.

Анализ текста состоит из нескольких задач:

* аннотация к документам,
* соответствующее представление документа,
* уменьшение размерности для обработки алгоритмических вопросов
* соответствующая функция классификатора для получения обобщения.

Процесс предварительной обработки состоит в том, чтобы очистить границу каждой языковой структуры и максимально устранить языковые факторы, зависящие от языка.

Представление документов является одним из методов предварительной обработки, который используется для уменьшения сложности документов и облегчения их обработки, документ должен быть преобразован из полнотекстовой версии в вектор документа. Текстовое представление является важным аспектом в классификации или кластеризации документов.

Текстовый документ обычно представляется в виде вектора весов терминов (словарных признаков) из набора терминов (словаря), где каждый термин встречается хотя бы один раз в определенном минимальном количестве документов. Основной характеристикой проблемы классификации/кластеризации текста является чрезвычайно высокая размерность текстовых данных. Количество потенциальных возможностей часто превышает количество документов. Определение документа состоит в том, что он состоит из терминов, которые имеют различные шаблоны возникновения. Предварительная обработка включают в себя такие этапы как: извлечение признаков (характеристик) и выбор признаков.

Извлечение признаков является первым этапом предварительной обработки, который используется для представления текстовых документов в формате слов.

*Токенизация*‒ лексический (семантический) анализ текста, который находит минимальную единицу в тексте. Минимальная единица – это токен. В лексическом анализе токеном может быть одно слово, предложение или абзац.

*Частота термина в документе* ‒ в семантическом анализе часто используют алгоритм счетчика слов, чтобы найти частоту каждого слова в данном тексте. Распределение частот слов является фундаментальным фенотипом языка.

*Стеммитизация*– это преобразование морфологических форм слов в его корень, при помощи удаление окончание морфологических преобразований. Корень не обязательно должен быть существующим словом в словаре, но все его варианты должны соответствовать этой форме после завершения алгоритма.

После извлечения признаков важным шагом предварительной обработки текста является выбор признаков для построения векторного пространства, что повышает масштабируемость, эффективность и точность текстового классификации или кластеризации.

Основная идея выбора признака заключается в выборе подмножества объектов из исходных документов. Выбор признаков выполняется путем сохранения слов с наивысшим весом в соответствии с заданной мерой важности слова. Выбранные функции сохраняют первоначальный смысл и обеспечивают лучшее понимание данных и процесса анализа. Для классификации или кластеризации текста основной проблемой является высокая размерность пространства объектов.

## **Анализ существующих программных средств для извлечения именованных сущностей из текстов на естественном языке**

В ходе изучения состояния исследований в области извлечения именованных сущностей были рассмотрены следующие программные средства:

* General Architecture for Text Engineering (GATE) – система, разработанная в Университете Шеффилда и предназначенная для обработки естественного языка. Система применима для решения задач выявления смыслового содержания текстов на естественном языке и его упорядочивания в структурированной форме с помощью аннотирования отдельных сегментом текста. GATE может использоваться для анализа кореферентности документов, семантической аннотации, работы с онтологиями, машинного обучения. Инструмент GATE, как и программные модули для него, написан на языке Java. Есть возможность обработки различных форматов текстовых документов: простой текстовый, HTML, DOC и други. Для описания правил в системе GATE применяется язык Jape. К минусам системы можно отнести архитектуру обработки естественных языков. На практике архитектура системы не подходит для обработки гибких языков наподобие русского, немецкого или латинского, так как GATE не имеет возможности обрабатывать морфологию. К плюсам системы можно отнести свободу, которую предлагает язык правил Jape, работу с огромным количеством входных форматов, даже таких обычно «проблемных» для парсинга, как pdf и docx, кроссплатформенность за счет платформы Java, на которой система была разработана. Также важным преимуществом является открытость кода системы GATE.
* Pullenti SDK. Pullenti SDK – комплекс средств разработки от компании «ООО Семантик», предназначенный для работы с неструктурированными данными. Pullenti SDK применим для решения задач анализа текста и извлечения именованных сущностей из неструктурированных русскоязычных текстов. Выделение именованных сущностей основано на правилах. Однако для некоторых типов сущностей можно подгружать внешние словари (онтологии), содержащие описания существующих сущностей, тогда система при выделении именованных сущностей будет пытаться привязываться к внешним сущностям, описанным в онтологиях. Плюсом системы является то, что она бесплатна для некоммерческого использования.
* FreeLing – мультиязычная библиотека для обработки текстов. В инструменте содержатся два модуля для извлечения именованных сущностей – Basic-модуль и Bio-модуль. Первый, Bio-модуль, основан на машинном обучении. Он обеспечивает довольно высокую точность извлечения (свыше 90%), однако работает медленнее, чем Basic-модуль. Инструмент используется как библиотека, написанная на C/C++. В функционал, предоставляемый инструментом, входит разметка текста (токенизация), выделение предложений, морфологический анализ и т.д. Для работы FreeLing требуется наличие нескольких установленных внешних библиотек. Поддерживаемые платформы: GNU/Linux.
* Stanford NER. Инструмент для обработки естественных языков, основанный на машинном обучении. Для извлечения именованных сущностей в инструменте есть три встроенных класса, основанных на регулярных выражениях, и один класс, предназначенный для разметки токенов через модель. Инструмент разработан на языке Java. Существуют также API для работы с инструментом, написанные и на других языках программирования: Python, Ruby, Perl, F#/ C#/ .NET. Поддерживает только английский язык.
* ReVerb. Инструмент извлечения именованных сущностей, основанный на машинном обучении. На вход используемая в ReVerb нейросеть принимает простой текст, а результатом работы являются тройки вида: аргумент 1, связывающая фраза, аргумент 2. Инструмент разработан на языке Java. Приложение использует библиотеку анализа естественных языков OpenNLP для разметки текста по частям речи и выделения в нем именных групп. Поддерживает только английский язык.

Таким образом, сравнительный анализ программных средств показывает, что каждое из них обладает недостатками. Потому, наиболее подходящим вариантом для решения задачи классификации данных в рамках данной работы является разработка собственного модульного программного комплекса, который позволит легко масштабировать и дорабатывать необходимый функционал.

## **Реализация программного комплекса**

## **Инструменты реализации**

В качестве основного языка для разработки агрегатора данных был выбран Python – язык программирования общего назначения, поддерживающий объектно-ориентированное программирование. Python поддерживает структурное, объектно-ориентированное, функциональное и императивное программирование. Python был выбран по нескольким причинам.

*Лёгкость чтения*. Python невероятно легко читать, поэтому у программистов обычно не возникает проблем с пониманием кода, написанного их коллегами. Это делает общение между разработчиками в рамках одного проекта намного более эффективным. А наличие большое числа IDE для разработки веб-приложений на Python делает совместную работу еще проще.

*Хорошая визуализация.* Представление данных в интуитивно понятном формате в Python достигается с помощью различных графиков и диаграмм. Компании, занимающиеся веб-разработкой, используют библиотеки Python с возможностью визуализации данных (например, Matplotlib), чтобы создавать чёткие и простые для понимания неспециалистов отчеты.

*Встраиваемость и независимость от платформ.* Благодаря своей интерактивности и переносимости Python обладает хорошими возможностями для динамической семантики и быстрого прототипирования. Его можно легко встроить в широкий спектр приложений, даже в те, которые используют разные языки программирования. Поэтому с Python можно легко исправлять новые модули и расширять базовый словарный запас языка. Python, как C++, Java и другие высокоуровневые языки программирования, может работать с разными типами компьютеров, ОС и баз данных практически без модификаций.

*Асинхронное программирование.* Для написания и поддержки асинхронного кода Python не требуется много усилий, поскольку нет взаимных блокировок, конфликта данных или любых других сбивающих с толку проблем. Каждая единица такого кода выполняется отдельно от основного потока, что существенно повышает производительность и скорость отклика приложения. Повышенная эффективность разработки

*Гибкий подход.* Python имеет несколько парадигм и может поддерживать множество стилей программирования, включая процедурные, объектно-ориентированные и функциональные. Это делает Python отличным языком для стартапов, поскольку им может потребоваться изменить свой подход в любой момент.

*Упрощённая реализация ООП.* Объектно-ориентированное программирование (ООП) — это парадигма, которая объединяет различные поведения и свойства в несколько объектов и классов. У каждого из этих классов есть своя функция, поэтому если в какой-то части кода возникает ошибка, другие части не затрагиваются. В Python работа ООП значительно упрощена, что делает разработку менее затратной и трудоемкой.

*Богатая стандартная библиотека и экосистема.* Библиотеки Python содержат огромное количество заранее написанного кода. Таким образом, разработчикам не нужно тратить время на создание основных элементов. Эти библиотеки также позволяют программистам обрабатывать и преобразовывать данные, необходимые для непрерывной обработки данных в машинном обучении (ML).

*Интеграция с другими языками.* Популярность использования Python для корпоративных программных приложений, во многом объясняется его плавной интеграции с другими языками, традиционно применяемыми в корпоративной разработке, такими как Java, PHP и .NET. Python может легко соединять отдельные компоненты приложения, написанные на разных языках. Неудивительно, что его иногда называют «склеивающим языком» (glue language) или языком интеграции. Python делает прямые вызовы из/в кода Java, C ++ или C. Это позволяет обеспечить контроль большинства процессов и реализацию наиболее распространенных протоколов и форматов данных. Кроме того, его можно применять для сборки новых и старых фрагментов инфраструктуры, что является типичной задачей при разработке сложных мобильных приложений.

В качестве основного языка для разработки веб-интерфейса был выбран JavaScript – объектно-ориентированный язык программирования для написания сценариев. Таким образом, посредством языка JavaScript реализуется возможность программирования на стороне клиента. Язык JavaScript предоставляет возможность доступа к элементам разметки web-страницы посредством объектов. Java Script был выбран по нескольким причинам.

*Широкая распространенность.*Любой браузер, любая операционная система поддерживает этот язык. Не будет проблем с запуском, ни на стационарном компьютере, ни на мобильном устройстве. Разработчику сайта теперь не нужно писать отдельные версии для разных типов устройств, тестировать и бороться с неожиданными ошибками. Достаточно проверить работоспособность кода в наиболее популярных браузерах. Любые, даже малоизвестные, браузеры разрабатываются с возможностью выполнения скриптов.

*JavaScript-приложение не требует установки на компьютер пользователя.*Сценарии выполняются непосредственно в браузере, когда загружается страница. Пользователь не только не предпринимает каких-то действий для этого, он даже не замечает, что запустилось веб-приложение. Поэтому не требуется специальных знаний и навыков, чтобы использовать продукты разработки JS.

*Язык высокого уровня.*Это означает, что существует уже определенный набор команд, с помощью которых можно написать код. Не нужно прописывать действия на машинном коде. Язык высокого уровня упрощает работу программисту, но при этом не сужает спектр его возможностей.

*Быстрый для пользователя.*Преимущество скорости JS получает благодаря тому, что код не компилируется на стороне клиента, ведь скрипт выполняется в браузере пользователя. Это значительно уменьшает нагрузку на сервер по сравнению с веб-приложениями, написанными на других языках. Вся полученная информация выводится в понятном пользователю виде на Web-страницах.

В качестве серверной части приложения используется стек технологий Linux + Nginx + Mongo DB.

Для хранения данных проекта выбрана база данных Mongo DB. Основной сервис – СУБД класса NoSQL, позволяющая разработчикам приложений хранить и синхронизировать данные между несколькими клиентами. Поддержаны особенности интеграции с приложениями под операционные системы Android и iOS, что важно для реализации кроссплатформенности, реализовано API для приложений на JavaScript, Java, Objective-C и Node.js, что критически важно для работы с веб-интерфейсом приложения, а также возможно работать напрямую с базой данных в стиле REST из ряда JavaScript-фреймворков.

MongoDB — документоориентированная система управления базами данных с открытым исходным кодом. Для хранения данных используется JSON-подобный формат. Эта СУБД отличается высокой доступностью, масштабируемостью и безопасностью.

Главные особенности MongoDB:

1. *Документоориентированная база* — сохранение данных в формате документов вместо формата реляционного типа, это делает MongoDB очень гибкой и адаптируемой к бизнес-требованиям. Возможность хранения разных типов данных особенно важна при работе с большими данными, которые собираются из разных источников и не ложатся в одну структуру.
2. *Специальные запросы* — MongoDB поддерживает поиск по полям, диапазонные запросы и поиск по регулярным выражениям. Могут быть сделаны запросы для возврата определенных полей в документах.
3. *Индексация* — можно создать индексы для улучшения производительности поиска в MongoDB. Любое поле в документе может быть проиндексировано. Это обеспечивает высокую скорость работы СУБД.
4. *Репликация* — эта СУБД может обеспечить высокую доступность с помощью наборов реплик. Набор реплик состоит из двух или более экземпляров MongoDB. Каждая реплика набора может выступать в роли первичной или вторичной. Первичная реплика — главный сервер, который взаимодействует с клиентом и выполняет все операции чтения/записи. Вторичные реплики сохраняют копию данных первичной реплики с помощью встроенной репликации. Если с первичной репликой что-то случилось, происходит автоматическое переключение на вторичную реплику, затем она становится основным сервером.
5. *Балансировка нагрузки* — MongoDB использует концепцию шардинга для горизонтального масштабирования с помощью разделения данных между несколькими экземплярами БД. Она может работать на нескольких серверах, балансируя нагрузку и/или дублируя данные, чтобы поддерживать работоспособность системы в случае аппаратного сбоя.
6. *Доступность* — MongoDB поддерживает все популярные языки программирования, ее можно использовать бесплатно open source решение.

В одном документе могут быть поля разных типов данных, данные не нужно приводить к одному типу. Основное преимущество MongoDB заключается в том, что она может хранить любые данные, но эти данные должны быть в формате JSON.

**Nginx** – мощный инструмент для развертывания веб-сервера, который при правильной настройке превосходит Apache. Области применения Nginx весьма обширны – от кэширования HTTP до создания инвертированного прокси-сервера. Сервер работает на ОС Unix-типа и был успешно протестирован на OpenBSD, FreeBSD, Linux, Mac OS X, Solaris.

**3 Разработка и тестирование классификатора текстов**

**3.1 Разработка программы**

Разработанная система состоит из четырех модулей:

* Модуль работы с источниками данных
* Модуль предварительной обработки текста
* Модуль оценки подготовленного текста
* Модуль обработки действий пользователя

*Набор данных*

Набор данных, который будет использован, состоит в общей сложности из более чем 2000 документов. Документы распределены по пяти тематикам: спорт, бизнес, политика, развлечения и техника.

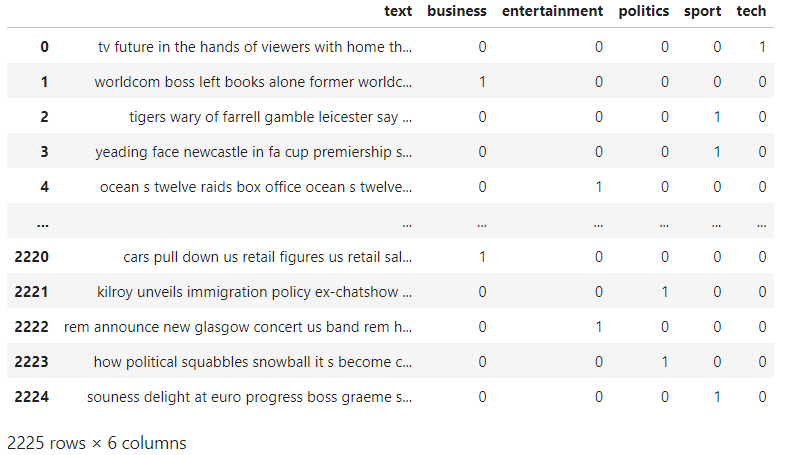


Рисунок 1 – Используемый набор данных

Модель классификации текста в Python реализует следующий алгоритм работы:

* импорт библиотек;
* импорт набора данных;
* предварительная обработка текста;
* преобразование слов текста в коэффициенты;
* обучающие и тестовые наборы;
* обучение модели классификации текста и прогноз;
* оценка модели;
* сохранение и загрузка модели.

*Импортирование набора данных*

Была разработана функция*create\_data\_set()* для импорта набора данных в приложение. Функция *create\_data\_set()* автоматически разделяет набор данных на данные и целевые наборы. Например, она будет рассматривать каждый файл внутри папки как одну категорию, и всем документам внутри этой папки будет присвоена соответствующая категория.



Рисунок 2 – Результат работы функции *create\_data\_set()*

*Предварительная обработка текста*

После импорта набора данных выполняется предварительная обработка текста. Текст может содержать цифры, специальные символы и ненужные пробелы. В зависимости от проблемы, решается задача удаления или не удаления этих специальных символы и цифр из текста.

Процедура начинается с удаления всех символов, не являющихся словами, таких как специальные символы, цифры и т.д. Затем удаляются все одиночные символы (союзы, сокращения слов). На следующем шаге данные преобразуются в нижний регистр, чтобы слова, которые на самом деле одинаковы, но имеют разный регистр, могли быть обработаны одинаково. Последним шагом предварительной обработки является лемматизация. При лемматизации слово сокращается до словарной корневой формы. Например, "cats" преобразуется в "cat". Лемматизация выполняется для того, чтобы избежать создания признаков, которые семантически похожи, но синтаксически различны.

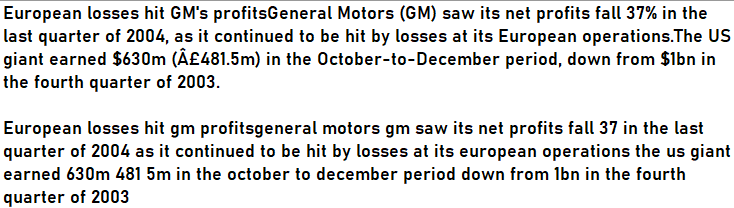


Рисунок 3 – Результат предварительной обработки текста

*Преобразование текста в коэффициенты*

Статистические методы, такие как машинное обучение, могут работать только с числами. Поэтому необходимо преобразовать обрабатываемый текст в числа. Существуют различные подходы к преобразованию текста в соответствующую числовую форму. Модель «мешка слов» и модель «вкрапления слов» — два наиболее часто используемых подхода. В данной работе используется модель «мешка слов» для преобразования текста в коэффициенты.

*Мешок слов*

При преобразовании слов в числа с помощью подхода «мешок слов» все уникальные слова во всех документах преобразуются в признаки. Все документы могут содержать десятки тысяч уникальных слов. Но слова, которые имеют очень низкую частоту встречаемости, не являются хорошим параметром для классификации документов. Поэтому включаются только те слова, которые встречаются не менее чем в пяти документах.

Слова, которые встречаются почти в каждом документе, обычно не подходят для классификации, поскольку они не предоставляют никакой уникальной информации о документе. И наконец, удаляются стоп-слова из текста, поскольку в случае анализа они могут не содержать никакой полезной информации.

Функция *fit\_transform()* класса *CountVectorizer* преобразует текстовые документы в соответствующие числовые признаки.

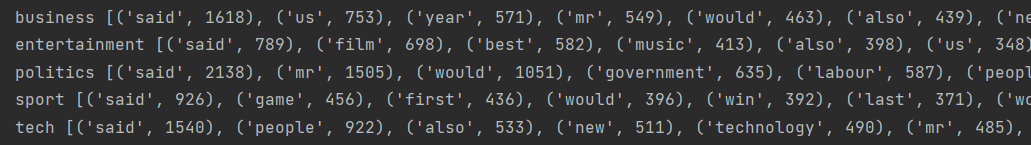


Рисунок 4 – Частота появления слов, распределенная по темам

*Нахождение TFIDF*

Использование мешка слов отлично подходит для преобразования текста в числа. Однако у него есть один недостаток. Он присваивает оценку слову, основываясь на его встречаемости в конкретном документе. При этом не учитывается тот факт, что слово может иметь высокую частоту встречаемости и в других документах.

TFIDF решает эту проблему путем умножения частоты термина слова на обратную частоту документа. TF означает "частота термина", а IDF — "обратная частота документа". Это своеобразная мера важности слова в контексте страницы, используемая поисковыми системами для оценки релевантности контента.

Эта мера определяет вес некоторого слова, который пропорционален частоте употребления этого слова в документе и обратно пропорционален частоте употребления слова во всех документах коллекции.

Это простой и удобный способ оценить важность термина для какого-либо документа относительно всех остальных документов. Принцип такой — если слово встречается в каком-либо документе часто, при этом встречаясь редко во всех остальных документах — это слово имеет большую значимость для того самого документа.

Применение технологии позволяет правильно рассчитывать коэффициенты встречаемых слов.

*Обучение модели классификации текста и прогнозирование*

Входные данные были разделены на обучающий и тестовый наборы. Будет использован метод Байеса (реализованный программно), который основан на анализе совместных распределений признаков документа и категорий. В задаче классификации текстов метод Байеса применяется отдельно для каждой категории, после чего принимается решение, принадлежит документ категории или нет. Метод Байеса обладает высокой скоростью работы и простотой математической модели. Этот метод часто используется в качестве базового метода при сравнении различных методов машинного обучения.

Наивный Байесовский классификатор – вероятностный классификатор, который делает предсказания, основываясь на списке определенных классов. Конечной задачей является определение к какому классу принадлежит документ, поэтому необходимо получить не саму вероятность, а наиболее вероятный класс. Для этого НБА производит оценку апостериорного максимума. Другими словами, рассчитывается вероятность для всех классов и выбирается тот, который обладает максимальной вероятностью.

Под моделью НБК понимается совокупность информации, которая вычисляется на основе данных, полученных во время обучения классификатора, а именно:

• общее количество документов, участвовавших в обучении;

• количество документов по каждому классу;

• количество уникальных слов в обучающем наборе;

• общее количество слов по каждому классу.

Для еще большего увеличения производительности алгоритма была использована концпеция параллельного программирования (асинхронная проверка принадлежности документа к каждой рубрике. Для этого были использованы стандартные модули python’a *queue* – для очередей и *threading* – для распараллеливания. Пройдя циклом по всем рубрикам, было выполнено наполнение очереди категориями *queue.put(category)* и запущены параллельные процессы обработки *threading*. В каждом таком процессе программа забирает рубрику *queue.get()* и начинает ее обработку, после завершения который осуществляется вызов *queue.task\_done()*, сигнализирующий о том, что обработка конкретной категории завершена. Чтобы синхронизировать потоки использовался вызов *queue.join()*, который ожидает завершение выполнения всех запущенных работ.

*Оценка модели*

Для оценки эффективности модели классификации можно использовать такие показатели, как матрица смешения (это таблица или диаграмма, показывающая точность прогнозирования классификатора в отношении двух и более классов) и точность предсказания. Чтобы найти эти значения, можно использовать утилиты classification\_report, confusion\_matrix и accuracy\_score из библиотеки sklearn.metrics.

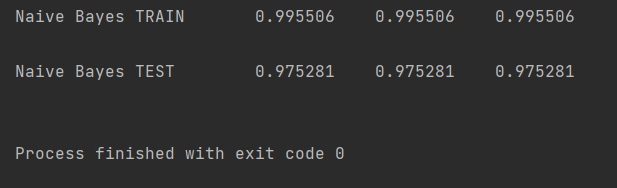


Рисунок 5 – Проверка точности работы классификатора

Из результата видно, что модель достигла точности 97,6%, что является отличным показателем.

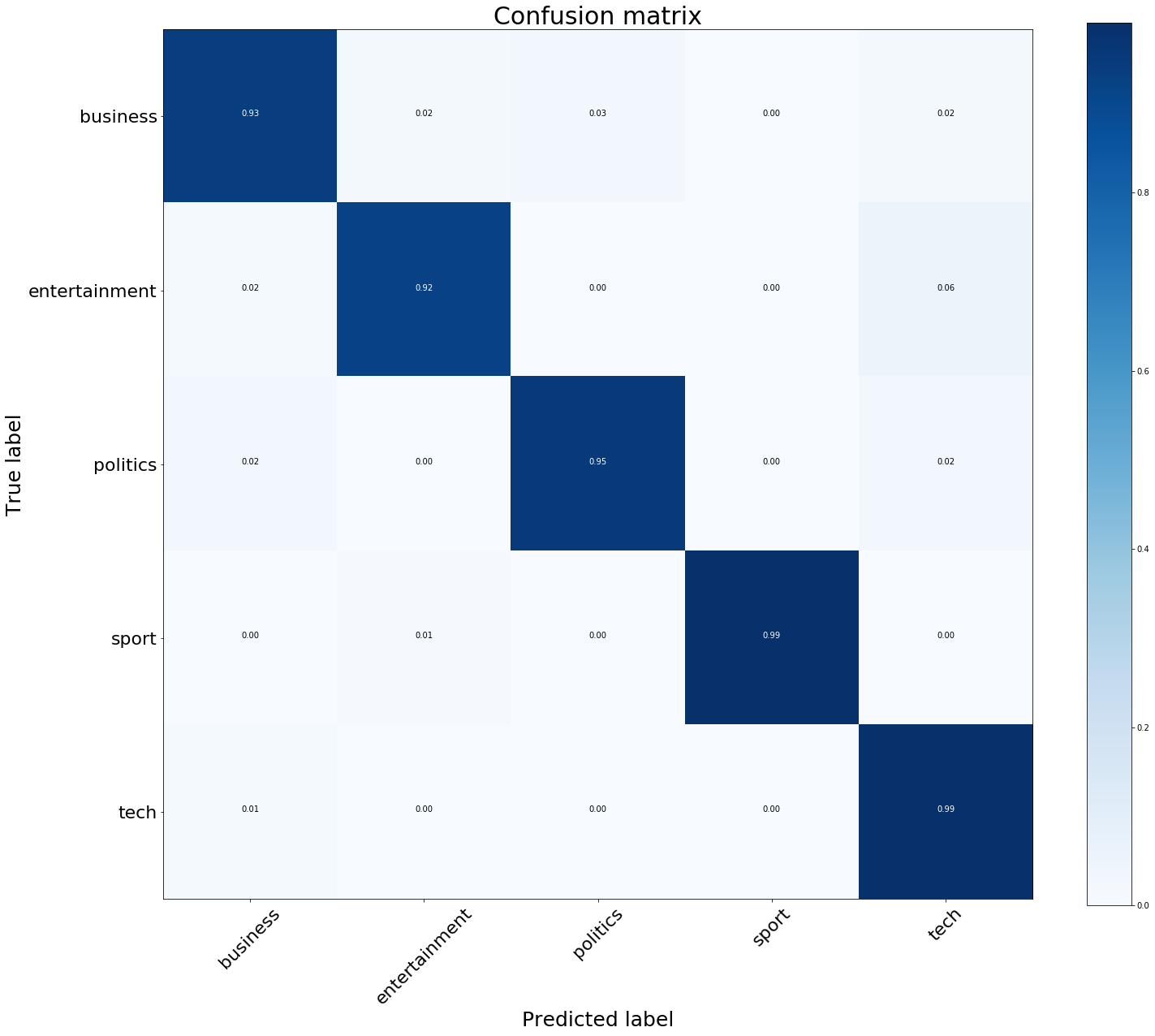


Рисунок 6 – Матрица смешения

Прогнозы классификатора находятся на оси X, а результат (точность) — на оси Y. Матрица смешения созданного классификатора показывает высокую точность обучения системы.

**3.2 Проверка работы программы с использованием случайных текстов.**

Для проверки работоспособности программного комплекса был использован API (совокупность инструментов и функций в виде интерфейса для создания новых приложений, благодаря которому одна программа будет взаимодействовать с другой) веб-сайта NewsCatcher.com, который позволяет запрашивать популярные новости различных тематик.



Рисунок 7 – Пример ответа от сайта NewsCatcher

Во время работы программы с сайта NewsCatcher запрашивается последние 5 новостей, для каждой из которых последовательно определяется её тема, после чего создается файл, в который записывается полученная новость, а сам файл сохраняется в папку с названием, соответствующим теме новости.

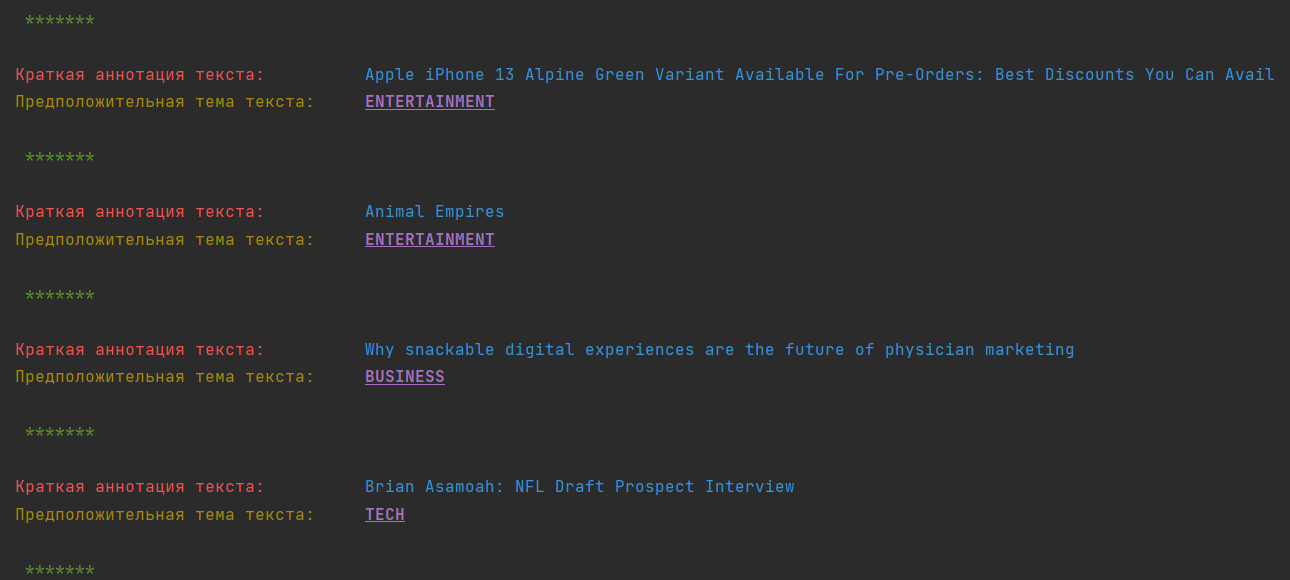


Рисунок 8 – Резльтат категоризации полученной информации

Кроме того, существует возможность добавлять любые текстовые файлы во входную папку (Input), откуда он будет автоматически считан и разобран программой.

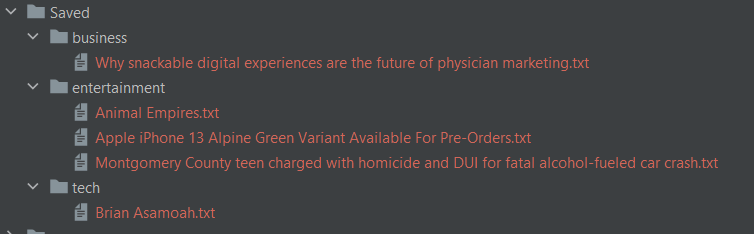


Рисунок 9 – Распределение полученных новостей по категориям в файловой системе компьютера

Результатом работы системы на тестовых примерах стали успешное добавление источников информации и интерпретация текстов по заданной тематике, а также последующая работа с полученным контентом. Программный комплекс показал себя полностью функциональным и готовым к работе. Апробация программного комплекса на нескольких источниках показала, что из 600 новостей у 586 из них темы были интерпретированы верно, что составляет приблизительно 97,7%.

Таким образом, точность работы комплекса достаточно высока. Проблема неверно извлеченных и интерпретированных тем может быть решена с помощью модерации базы ранее верно определенных тем автоматически, для чего разработанный программный комплекс предоставляет инструментарий.

# **ЗАКЛЮЧЕНИЕ**

Реализованный программный комплекс планируется использовать для решения задачи автоматического извлечения тем документов и структурирования данных из файлов на естественном языке, что позволит обеспечить оптимальную организацию процесса сбора информации, а также увеличит простоту поиска нужной информации.

В результате проделанной работы было разработано программное обеспечение, базирующееся на байесовском алгоритме, которое позволяет определять тематику текста на основе данных, полученных во время обучения классификатора. Также представлены подходы по оптимизации и увеличению скорости работы программы, такие как параллельное программирование и уменьшение числа термов. Проведен и проанализирован машинный эксперимент по обучению на основе статей различных тематик. НБА требует меньший объем обучающих данных, чем другие подобные алгоритмы и лучше работает с категорийными признаками.

В первой главе были проведены обзор современной библиографии, научных исследований в области агрегации данных и обработки естественного языка. Было проанализировано положение дел в области машинного обучения, рассмотрены основные направления анализа естественных языков и возможность их применения для решения поставленной задачи, методы извлечения именованных сущностей и их возможное применение в агрегации данных, алгоритмы парсинга данных на естественном языке, проведен сравнительный анализ программных средств для извлечения именованных сущностей из текстов на естественном языке.

Во второй главе были выбраны программные средства реализации системы, описаны их достоинства и недостатки, проанализированы функциональные требования к разрабатываемому программному комплексу, проведено проектирование приложения.

В третьей главе были разработаны модули и компоненты системы, которые в совокупности при взаимодействии друг с другом обеспечивают надежную и бесперебойную работу приложения в целом, удобство и комфорт работы с ним пользователя. Были разработаны модуль работы с источниками данных, модуль предварительной обработки текста, модуль оценки подготовленного текста, модуль обработки действий пользователя, посредством которого пользователь взаимодействует со всем комплексом. Были сделаны выводы о корректной работе разработанного приложения. В ходе апробации система была протестирована на 600 источниках, точность извлечения фактов из текстов на естественном языке составила 97,7%.

Таким образом, приложение позволяет производить классификацию текстов на естественном языке по пяти темам. В будущем планируется расширить библиотеку используемых тем с целью обеспечить охват более широкого спектра проблем, что позволит более точно реагировать на них.