Московский государственный университет имени М. В. Ломоносова

Факультет вычислительной математики и кибернетики

Кафедра алгоритмических языков

Протасов Александр Васильевич

Разработка системы категоризации текстов

Курсовая работа

Научный руководитель: доцент, к.ф.-м.н.

Волкова Ирина Анатольевна

Москва 2022

Аннотация

В работе рассматривается задача категоризации текстов с использованием методов машинного обучения. Исследованы подходы предварительной обработки текста и создана коллекция датасетов для обучения методов и вычислительных экспериментов. Проведено сравнение на следующих методах: логистическая регрессия, наивный байесовский классификатор, деревья решений.

Для каждого метода получена оценка качества работы. И на основе этого выбран наиболее эффективный метод: логистическая регрессия.

Содержание

1 Введение

Мы постоянно сталкиваемся с большим количеством текстовой информации, работаем с ней, и зачастую она представляется нам в виде неструктурированных данных. Отсюда возникает задача, когда этой информации хранится огромное количество: как не запутаться в ней, как ее представить в удобном формате так, чтобы было удобно ориентироваться, эффективно работать, и получать быстрый доступ к любой информации по ее категории или по каким-то признакам. Данную задачу называют задачей категоризация текстов - определение новых текстов в одну из предопределенных категорий.

Есть несколько подходов категоризировать тексты:
Первый способ - вручную соотносить тексты к соответствующим категориям. Это неэффективно, так как все больше возникает необходимость классифицировать большое количество текстов за короткие сроки. Еще один подход заключается в написании ряда правил, которые будут автоматически относить каждый поступающий на вход документ к одной из предопределенных рубрик. С другой стороны, в мире технологии достигли таких высот, что многие задачи, которые делает человек, могут выполнять вычислительные машины, и в основном более эффективно. Поэтому все чаще с помощью них автоматизируют рутинные и трудоемкие задачи, и в том числе затрагивающие обработку текстов. Как раз третий способ об этом. Он задействует алгоритмы машинного обучения. Классификация текстов, основанная на машинном обучении, является примером обучения с учителем: человек задает набор классов, размечает обучающие тексты и уже на новых данных машина предсказывает категорию.

В своей работе будет задействован третий способ, и с точки зрения машинного обучения данная проблема рассматривается как задача классификации -- предсказание, к какому классу относится объект [1].

Для следующих прикладных задач можно применить автоматическую классификацию:

1. Борьба со спамом. Здесь идет бинарная классификация писем: “спам”, “не спам”.
2. Разбиение новостей по разным категориям: спорт, технологии, красота…
3. Подбор контекстной интернет-рекламы - определить тему сайта или запроса пользователя для соответствующей рекламы.

Мною была прочитана статья [2], в которой рассказывается о проекте “FOSS News” – это проект, который занимается составлением дайджестов посредством классификации по ключевым словам. Познакомившись с данной тематикой и пермским коллективом данного проекта, они предложили реальную задачу – классификация веб-страниц по категориям на основе текста материала. Также они предоставили мне исходные данные – множество веб-страниц, для каждой из которых проставлена одна из предопределенных категорий.

https://0x1.tv/Дайджесты_FOSS_News_и_средства_автоматизации_их_составления_(Дмитрий_Винокуров,_OSSDEVCONF-2021)

!!!ПОМЕНЯТЬ НОМЕРА СТРАНИЦ

2 Постановка задачи

Целью курсовой работы является решение задачи категоризации текстов методами машинного обучения: по имеющемуся набору веб-страниц требуется разработать методы сбора информации с веб-страниц, предварительной обработки текстов, формирования датасетов, на которых будет произведено сравнение результатов классификаторов и выбор оптимального из них.

Для достижения поставленной цели необходимо выполнить следующее:

1. Изучить существующие подходы категоризации текстов различных авторов, их преимущества и недостатки, методы обработки текстовой информации на веб-страницах, методы классификации в машинном обучении.
2. Разработать и реализовать алгоритм, позволяющий получать текстовые данные с веб-страниц.
3. Подготовить исходные данные для решения поставленной задачи.
4. Сравнить полученные результаты с результатами существующих решений задачи.
5. Выбрать наиболее оптимальный метод для системы категоризации текстов.

3 Обзор существующих подходов к решению задачи

Для решения данной задачи NLP предлагается использовать элементы машинного обучения. Ввиду этого здесь рассматриваются решения различных авторов с соответствующими алгоритмами.

В работе [2] в качестве категоризации алгоритма используется метод монотонного ближайшего соседа. Он сравнивается с методами ближайшего соседа и бустинга. Для экспериментов был взят популярный набор данных на английском языке, состоящий из 20 новостных категорий. Обучающая и тестовая выборка отличались по количеству, словарь из 8000 слов. Для сравнения и получения результатов выделено 100 признаков. Было показано, что с данным корпусом метод монотонного ближайшего соседа работает лучше и достигает 97% правильности.

В работе [3] задача решается для классификации документов научно-образовательных организаций. Представлена своя система признаков документов. Рассматривалось 6 методов машинного обучения, по которым зафиксированы результаты. Классификаторы сравнили до и после предварительной обработки данных. А в качестве показателей взяли время обучения классификатора и точность классификации. Результаты значительны: по времени достигается трехкратное улучшение, увеличение точности варьируется от 5 до 20%.

В работе [4] для классификации текстов используется метод опорных векторов (SVM). При избежании многомерных входных пространств автором исследуется идея о том, что большинство признаков не имеют значения. Но он показал, что это не так. Все признаки (одной из категорий) были ранжированы в соответствии с их приростом информации. Затем наивный байесовский классификатор обучается, используя признаки, которым присвоен рейтинг 1-200, …, 2001-4000, 4001-9962. Результаты показали, что даже признаки с низким “рейтингом” содержат значительную информацию. SVM классификатор с различными ядрами (полиномиальное, радиальная базисная функция) тоже сравнили с другими классификаторами. Метод опорных векторов с радиальной базисной функцией достиг наилучшего результата.

В работе [5] задача рубрикации документов поставлена как задача автоматической категоризации веб-страниц. Отличаются они одним этапом: во втором случае вначале из веб-страницы извлекается текст. Классификатором, который имел самую высокую точность, является метод опорных векторов с точностью 0,931. Для него были показаны 15 основных признаков по каждой категории, а также пример с несколькими словами и их весом для всех категорий (например, слово «археология» имеет наибольший вес категории 1 по сравнению с другими, что показывает его релевантность для категории 1).

Были взяты разные подходы к решению общей задачи - категоризации текстов. В том числе было важно посмотреть на то, как категоризация текстов используется для реальных случаев: задействование в организации, обработка "сырых" данных (взятых с веб-страниц или приложений). Для сравнения изучено и решение, где используются готовые датасеты, т.е. в учебных целях. Также интересно изучить, как представляются у авторов результаты в проведении экспериментов, на что они обращают внимание. Все выделенные мной ключевые моменты описаны в данных работах.

4 Подготовка данных

Для начала нужно создать набор данных в удобном формате, чтобы можно было работать с данными напрямую. Потом, как и в любой задаче NLP, необходимым процессом является преобразование текста в вид, который методы машинного обучения могут считывать. А также формирование данных в удобный для человека формат, чтобы можно было легко работать с ними.

Основные этапы подготовки данных для машинного обучения включают в себя:

- Обработка текста
    - Токенизация и применение правил преобразования исходных текстов
    - Лемматизация, стемминг
    - Удаление стоп-слов
- Представление набора документов векторами
- Уменьшение размерности пространства признаков

4.1 Описание и подготовка данных

Набор данных, предоставленный изначально пермским коллективом разработчиков, содержит 5007 записей на русском языке и 3632 - на английском. Далее работа будет проводиться только с данными на русском языке. Записи поделены в разном количестве между 26 категориями (Рис. 1).

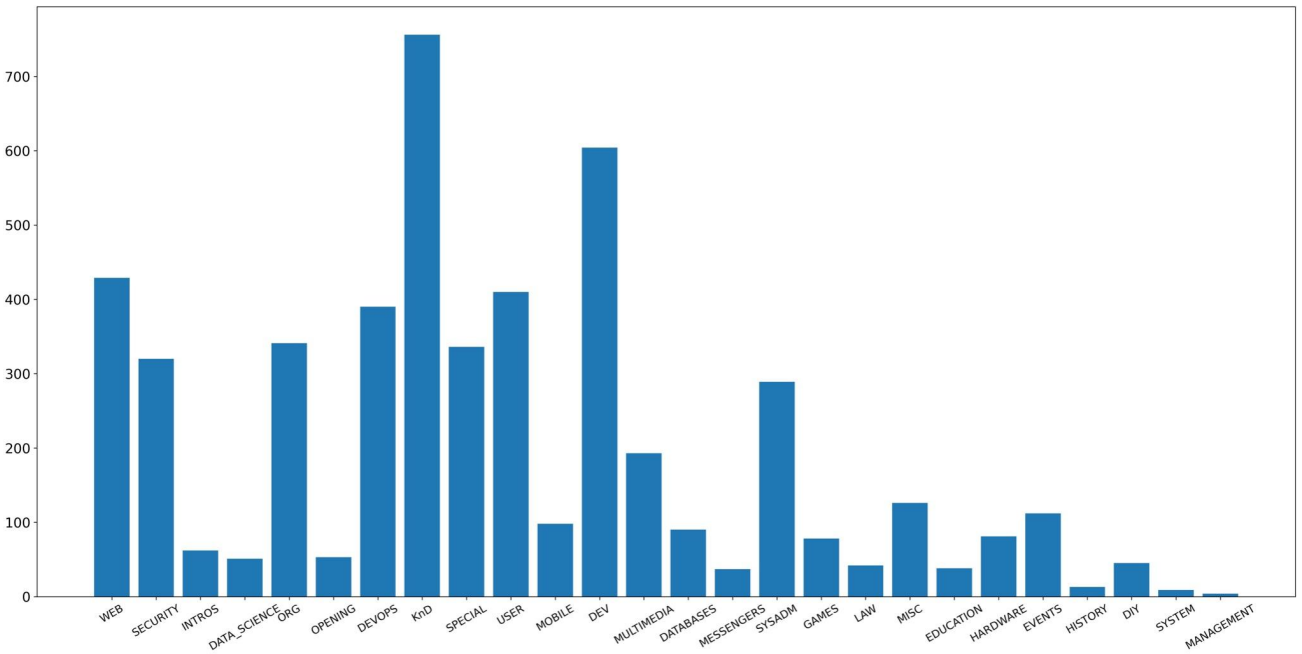

Рис. 1. Количество записей каждой категории

Чтобы было удобно оперировать с документами, создан класс с записями, где каждая содержит, как минимум, поля text и isDownloaded. Второе поле нужно для того, чтобы учитывать, что веб-страница может быть недоступна. В связи с этим параметром получилось 4905 документов. Все эти записи сохранены в файл **russianRecords.json**.

Для обучения классификаторов и последующих сравнений результатов будет задействовано несколько датасетов, созданных мной из предобработанного выше набора данных. Чтобы знать, как корректно разбить на датасеты, надо, как минимум, посмотреть еще раз на данные (Рис. 1). По рисунку видно, что есть категории, которые отличаются количеством записей между собой в 10 и более раз, говоря другими словами, присутствуют миноритарные и мажоритарные классы.

Возникает проблема с несбалансированными данными. Это может плохо сказаться на результатах: некоторые классы могут не иметь достаточного веса для воздействия на классификатор. Одним из решений этой проблемы является субдискретизация (under-sampling) – выбор случайного подмножества мажоритарного класса для создания более сбалансированного набора. [6]

На качество работы модели также влияет размерность релевантных признаков (более обширно - количество документов), которые могут характеризовать ту или иную категорию. Отсюда возникает необходимость рассмотреть разное количество классов в датасетах.

Учитывая все это, у меня получилось сформировать 5 оптимальных датасетов в формате .**csv**:

- **dataset_75_15** – 15 категорий, для каждой из которых дано 75 документов
- **dataset_150_10** – 10 категорий, для каждой из которых дано 150 документов
- **dataset_200_15** – 15 категорий, для каждой из которых дано либо 200 документов (если категория имеет большее количество документов), либо свое число документов (иначе)
- **dataset_200_23** – 23 категории, для каждой из которых дано либо 200 документов (если категория имеет большее количество документов), либо свое число документов (иначе)
- **dataset_350_4** – 4 категории, для каждой из которых дано 350 документов
- **dataset_max_23** – 23 категории, для каждой из которых дано максимум из числа документов

4.2 Обработка текста

Текст в NLP определяется как последовательность слов или в общем виде токенов, а слово – как значимая последовательность символов. Границы токенов в русском языке можно найти, например, с помощью пробелов или знаков препинания.

Пусть на вход подаются все тексты, взятые с веб-страниц, – это корпус. По окончании предварительной обработки корпуса мы должны получить словарь - набор всех уникальных токенов. Под "токеном" в разных задачах можно считать как не одно словосочетание, так и символ. Зафиксируем, что в данной задаче токен - слово.

4.2.1 Токенизация и правила преобразования исходных текстов

Основой других этапов обработки текста являются токенизация и применение правил преобразования текстов. Правила нужны, чтобы удалить специальные символы (например, @ $ #), знаки препинания, одиночные цифры, тэги html <>. Но можно ли улучшить процесс извлечения токенов? То есть уменьшить выходной размер словаря и убрать неинформативные слова (слова на английском языке, слова содержащие цифры).

Например, если разбиение токенов происходит по пробелам и любым знакам препинания, то необязательно, чтобы точка являлась концом предложения. Она может встречаться и в сокращении, и в дате.

Следующее, что нужно учесть, – регистр слов. В словаре могут содержаться два одинаковых слова, но только одно начинается с большой буквы, а другое – с маленькой. Решением данной проблемы является преобразование всех символов в строчные. Поэтому слова "стэк", "Стэк", "стЭК" соответствуют одному и тому же токену.

Для обработки подобных случаев понадобится ряд рукописных правил. Их можно представить в виде регулярных выражений [7]. С помощью них задается шаблон, который найдет во всем тексте то, что соответствует ему, и заменит на указанное. Как пример: все слова, содержащие хотя бы одну цифру, заменить на пустую строку.

4.2.2 Лемматизация и стемминг

Другая подзадача, которую надо решить, – убрать многообразие форм слова. Есть два частных случая нормализации слов: лемматизация и стемминг [8].

Лемматизация - это процесс приведения словоформы к лемме, то есть начальной форме слова.

Стемминг - это процесс отсечения от слова окончаний, то есть приведение слова к стемме (основе).

Стемминг в частности основан на правилах (например, алгоритм усечения окончаний) и требует разбиения слова на подстроки, поэтому выполняется быстрее. Но в тоже время не для всех частей речи можно сформировать хороший набор правил. Лемматизация в свою очередь включает теги частей речи и применение для каждой части речи своих правил нормализации. То есть лемматизация более точна, но она медленнее обрабатывается, так как ей приходится просматривать словарь [6].

4.2.3 Удаление стоп-слов

Следующий этап, с помощью которого можно избавиться от неинформативных слов, – удаление стоп-слов – часто встречающиеся слова, которые не несут существенной смысловой нагрузки (союзы, предлоги, частицы и другое). Они добавляют только шум в данные.

Есть два основных метода [9]:

- Завести список стоп-слов. Его можно пополнять со временем или зафиксировать для конкретной задачи.
- Удалить слова, которые встречаются слишком часто.

4.3 Представление набора документов векторами

По завершении предварительной обработки текст нужно перевести в числовое представление, которое уже можно применить к алгоритмам машинного обучения. Или продолжить анализ выделения признаков для сокращения входного пространства.

“Мешок слов” (Bag of words)

Один из простых способов подготовить текст для машинного обучения – это векторизация (конвертация текста в числа) текста с помощью “мешка слов”.

Этот метод представляет текст в виде мультимножества его термов (выделенное нормированное слово), каждому из которых ставится в соответствие некоторый вес w. Значит, каждый текст (документ) d из корпуса определяется как числовой вектор:

$$d = (w_1, w_2, ..., w_{n-1}, w_n),$$

где n - размер словаря, являющийся множеством всех термов из всех текстов (D). Тогда “мешок слов” можно представить в виде матрицы, где столбцы – слова из словаря, а каждая строка – определенный документ d.
Определить значимость (вес) w в тексте можно несколькими способами:

- бинарная частота:

$$w = \begin{cases} 1, \text{ если слово не встречается в тексте} \\ 0, \text{ если слово встречается в тексте} \end{cases}$$

- TF (term frequency – частота терма) измеряет, насколько часто некоторый терм t встречается в тексте. Она выражается формулой:

$$w = TF(t, d) = \frac{n_t}{\sum_k n_k}$$

  где n_t– количество употреблений t в документе d, а в знаменателе – общее число термов в данном документе.

- TF-IDF (inverse document frequency – обратная частота документа) – мера, с помощью которой оценивается важность слова в документе, являющегося частью корпуса.

  IDF показывает, что если терм употребляется во всех документах, то он является неинформативным. То есть так можно показать, насколько уникально слово в D.

IDF для каждого терма t определяется по формуле:

$$IDF(t, D) = log \frac{|D|}{\left|\{d_i \in D | t \in d_i\}\right|}$$

где $|D|$ – размер корпуса, а в знаменателе - количество документов, в которых содержится терм t.

Тогда значение TF-IDF для терма t в документе d вычисляется как:

$$w = TF - IDF(t, d, D) = TF(t, d) * IDF(t, D)$$

Значит, больший вес будет у тех слов, которые употребляются в тексте чаще, чем во всем остальном корпусе.

"Мешок слов" не учитывает грамматику и порядок слов в документе. Данные проблемы можно решать с помощью **N-грамм** – последовательность из n элементов (в частности, слов или букв). Поэтому если важно сохранить последовательность слов, выявить какую-то закономерность: рассчитать вероятность, какое слово чаще следует за заданным, можно использовать комбинацию из нескольких слов, например, биграммы (N = 2).

4.4 Уменьшение размерности пространства признаков

Причина, по которой нужно задействовать методы снижения размерности, - некоторые объекты пространства содержат избыточную информацию и несут шум, что может сказаться на увеличении временных затрат, а также для методов машинного обучения может снизиться их точность. И за счет снижения размерности пространства надеюсь, что получится выделить и сохранить наиболее важные объекты пространства, а другие - отбросить.

Токен, который используется только в одном документе, скорее всего, не встретится в тестовом наборе, и хранить его в словаре не имеет смысла, он излишен. Эту проблему можно решить следующим подходом – оставить только такие токены (признаки), которые встречаются, как минимум, в k документах [9]. Где $k = [2, |D|]$.

Также можно удалить слова, которые встречаются часто и поэтому менее важны для классификации. Это можно сделать, например, с помощью обратной частоты IDF.

Методы, которые автоматически сохраняют самые важные признаки (размерность удается уменьшить автоматически, если выбрать лучшие признаки): рекурсивное устранение признаков (RFE), последовательный отбор признаков. Эти алгоритмы учитывают внешнюю оценку, которая присваивает веса признакам (например, для дерева решений - это важность признаков).

Рекурсивное устранение признаков

Цель данного метода - выбрать признаки путем рекурсивного рассмотрения меньшего и меньшего наборов признаков. Этот алгоритм вначале обучает модель начальному набору признаков и вычисляет производительность модели. Затем алгоритм отбрасывает, в зависимости от коллинеарности между признаками и их важности, по k признаков каждый раз. Это происходит до тех пор, пока не будет достигнуто указанное модели количество признаков. [10]

Последовательный прямой отбор признаков

Данный алгоритм вначале обучает каждый признак отдельно из набора данных и выбирает лучший. Затем алгоритм добавляет или удаляет один элемент за раз, в зависимости от оценки производительности классификатора, до тех пор, пока не будет достигнут желаемый размер признаков. [11]

5 Методы машинного обучения

Выбор метода машинного обучения является также важным этапом. Он будет применяться к векторному представлению документов и с его помощью может быть построен алгоритм, способный распознавать разные категории текста.

В настоящее время существует большое количество методов машинного обучения, которые разработаны для различных задач и которые отличаются между собой реализацией и используют разного рода математический аппарат.

Ранее были рассмотрены решения задачи категоризации текстов [2, 3, 4, 5], авторы которых используют различные методы машинного обучения:

- метод ближайшего соседа
- логистическая регрессия
- дерево решений
- метод Байеса
- метод опорных векторов

В работе далее будут рассмотрены распространенные методы машинного обучения, для которых в том числе опубликованы результаты их применения для задачи - категоризация документов. Из вероятностных методов - метод Байеса, из логических - дерево решений, из линейных - логистическая регрессия [9].

5.1 Логистическая регрессия

Логистическая регрессия является линейным методом классификации. Этот метод используется для прогнозирования вероятности некоторого события (категория документа) по значениям множества признаков [12].

Чтобы понять основную идею данного метода, нужно рассмотреть его для бинарной классификации.

Пусть дана выборка - $(x_i, y_i), i = \overline{1, m}$ состоящая из m объектов (документов), каждый из которых описывается n признаками (токенами) $x_i \in \mathbb{R}^n$ и принадлежит одному из двух классов $y_i \in \{0, 1\}$.

Цель: построить функцию такую, чтобы прогнозируемый ответ $\hat{y}_i \in [0, 1]$ был как можно ближе к фактическому ответу y_i.

Как раз структуру такой функции задает логистическая регрессия [14]:

$$\hat{y}_i = f(x_i, w, b) = \sigma(w^T \cdot x_i + b),$$

где $w \in \mathbb{R}^n, b \in \mathbb{R}$ – параметры модели, оцениваемые в ходе обучения, которые также называются просто коэффициентами.

То есть логистическая регрессия для объекта x_i получает предсказание в 2 шага:

1) преобразование из вектора в число ($z \in \mathbb{R}$):

$$z = \sum_{j=1}^{n} w_j \cdot x_j + b$$

2) теперь число z преобразуется в меру вероятности с помощью функции активации - сигмоида (Рис. 1):

$$a = \sigma(z) = \frac{1}{1 + e^{-z}}$$

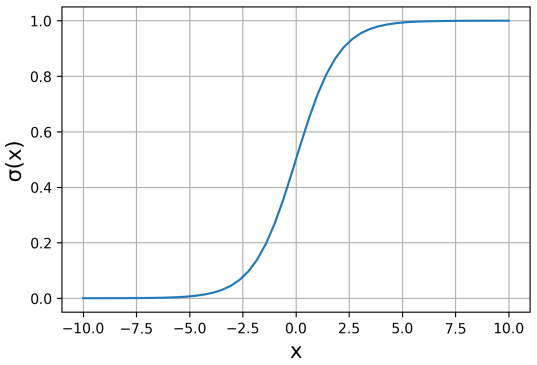

Рис. 2. Функция сигмоида

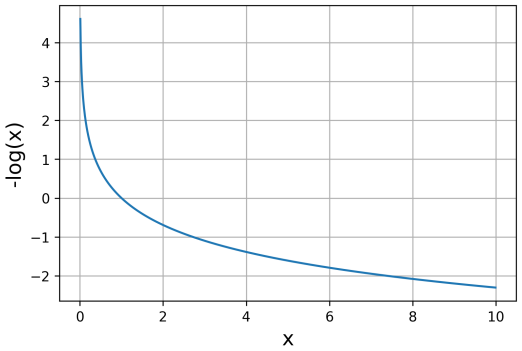

Рис. 3. Функция логарифм

Для подбора двух неизвестных параметров w и b нужно воспользоваться стандартным, для машинного обучения, способом – путем минимизации функции потерь.

Введем функцию потерь [15]. Для логистической регрессии потеря на объекте y_i, если мы предсказываем $\hat{y}_i$, определяется как:

$$L_i(\hat{y}_i) = -y_i log(\hat{y}_i) - (1 - y_i) log(1 - \hat{y}_i)$$

Когда $y_i = 1$, L_i для соответствующего предсказания $\hat{y}_i$ равна $-log(\hat{y}_i)$. И если предсказывается вероятность близкая к 1, то это тот результат, который и нужен,

т.е. небольшая потеря. Если иначе - предсказывается вероятность близкая к 0, то потери будут большими, так как $-log(\hat{y}_i)$ значительно падает (Рис. 3).

Теперь вспоминая про выборку из m объектов, можно определить функцию потерь на этой выборке:

$$L = \frac{1}{m}\sum_{i=1}^{m} L_i(\hat{y}_i) \to \min_{w,b}$$

То есть задача свелась к задаче минимизации функции потерь. Минимизация данной функции нужна для того, чтобы ошибка на обучающей выборке была минимальна.

Минимизация происходит с помощью градиентного спуска. Надо идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом. Шаг будет следующим (для b аналогично):

$$w_{j+1} := w_j - \alpha\frac{\partial L}{\partial w}(w_j, b_j)$$

где α – скорость градиентного спуска, производная функции потерь для одного объекта по параметру w: $\frac{\partial L_i}{\partial w} = \frac{\partial L_i}{\partial a}\cdot\frac{\partial a}{\partial z}\cdot\frac{\partial z}{\partial w} = (a - y_i)\cdot x_i$, а по параметру b: $\frac{\partial L_i}{\partial b} = (a - y_i)$, учитывая следующее:

$$L_i(w,b) = L_i(a(z(w,b))) = -y_i log(a) - (1 - y_i)log(1 - a),$$

$$\frac{\partial L_i}{\partial a} = -\frac{y_i}{a} + \frac{1 - y_i}{1 - a} = \frac{-y_i + a}{a(1 - a)}, \frac{\partial a}{\partial z} = a(1 - a), \frac{\partial z}{\partial w} = x_i \in \mathbb{R}^n.$$

Рассмотрев бинарную логистическую модель, можно через нее теперь прийти к логистической регрессии, которая решает задачу многоклассовой классификации. Для этого надо разделить многоклассовый набор данных (пусть K классов) на K независимых бинарных моделей логистической регрессии. То есть каждый класс представлен своим классификатором, который предсказывает, является ли наблюдение этим классом или нет. Такой метод называется “Один ко всем”(англ. “One Vs. Rest”) [8].

5.2 Дерево решений

Деревом решений называют ациклический граф, по которому производится классификация объектов, заданные множеством признаков. Данный метод часто используется в задачах классификации [12]. В задаче категоризации текстов объектами являются тексты, целевым атрибутом – их категории.

Обычное дерево включает в себя корень, ветви и листья. Также задается и бинарное дерево решений. Оно имеет корневой узел, являющийся родителем всех остальных узлов, которые в свою очередь соответствуют проверке свойства; ветви, показывающие результат проверки; листья, представляющие метки класса.

Теперь стоит более подробно рассмотреть алгоритм создания обучающего дерева решений [15, 16].

- Дерево начинается со своего корня. В нем изначально хранится весь набор данных (в данной задаче - это множество документов, каждый из которых задан числовым вектором размерности n). Для корня выбирается наиболее подходящий признак и наилучшее пограничное значение, задающее бинарное условие. Вследствие чего будет два потомка, в которые данные отправляются по следующим правилу: все документы, для которых условие истинно, переходят в левый узел, а документы, для которых условие ложно, - в правый узел.

- Далее продолжается процесс разделения - последующие узлы рекурсивно разделяются на более мелкие подмножества в соответствии с пограничным значением выбранного признака. Условия разделения автоматически выбираются на каждом шаге с учетом того, какое условие наиболее подходящим способом разделяет текущий набор документов [15]. Для того, чтобы измерить качество разбиения, существует несколько функций. Первая – основана на информационной энтропии, вторая -- неопределенность Джини. Далее речь пойдет про вторую метрику.

Данный показатель определяет, как часто случайно выбранный пример будет распознан неверно. Коэффициент Джини задается следующей формулой [17]:

$$Gini(Q) = 1 - \sum_{i=1}^{n} p_i^2$$

где Q - результирующее множество, n - число классов в нем, p_i - вероятность i-го класса.

Коэффициент Джини равен 0, если все примеры (документы) набора относятся к одному классу. Значит, чем меньше коэффициент Джини, тем меньше вероятность того, что выбранный пример в множестве будет классифицирован неправильно.

- Следующий шаг - определение остановки алгоритма. Существует несколько способов (в том числе тех, которые помогают при переобучении):
    - Построение полного дерева – когда все листья дерева являются однородными, то есть когда каждый лист содержит примеры, принадлежащие одному и тому же классу.
    - Ограничение глубин дерева – когда заранее задается максимальное число разбиений в ветвях, по достижении которого построение дерева завершается.
    - Определение минимального числа примеров в узле – когда любой узел имеет число примеров меньше заданного, он дальше не может создавать узлы.

- В завершении алгоритма выводится структура дерева.

Так, в процессе классификации осуществляются переходы сверху вниз между внутренними узлами дерева решений на основе условий. Если алгоритм дошел до конечного узла, классификация считается законченной. Таким образом, путь от корня до конечного узла - конъюнктивное правило, а все дерево - группа правил дизъюнктивного выражения.

К деревьям решений часто присуще следующие факты [6, 18] :

– Одиночные деревья решений обычно более подвержены проблеме переобучения (overfitting). И для уменьшения их сложности и увеличения качества работы существует алгоритм “обрезки деревьев” (pruning).

– Данный метод - аналог мышления на человеческом уровне. Обычно можно сделать несколько хороших интерпретаций.

– Деревья решений относят к категории “жадных алгоритмов”. То есть если было выбрано условие, по которому производилось разбиение, то алгоритм не может вернуться назад и выбрать другое условие.

5.3 Наивный байесовский классификатор

Метод Байеса [6,19] относится к вероятностным методам классификации. В его основе лежит теорема Байеса (пусть $c \in C$):

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

где $P(c|d)$ – вероятность того, что документ d соответствует классу c, $P(d|c)$ – вероятность найти документ d среди всех документов класса c, $P(c)$ — априорная вероятность найти документ класса c в корпусе документов; $P(d)$ — априорная вероятность документа d в корпусе документов.

Задача данного классификатора - найти более вероятностный класс. Для этого считаются вероятности всех классов и выбирается тот класс, который имеет наибольшую вероятность. Также $P(d)$ не зависит ни от какого класса из C, то есть это константа, которую можно игнорировать. Учитывая это, можно записать следующую формулу:

$$c_{max} = \underset{c \in C}{\operatorname{argmax}}[P(d|c)P(c)]$$

При представлении документа в виде вектора признаков ($d = (w_1, , w_n)$), делается “наивное” предположение о том, что все координаты независимы, то есть:

$$P(d|c) = P(w_1, w_2, ..., w_n|c) = P(w_1|c)P(w_2|c)...P(w_n|c) = \prod_{i=1}^{n} P(w_i|c)$$

Иначе говоря, опускается тот факт, что в тексте появление двух разных токенов часто взаимосвязано.

Тогда данное вычисление можно подставить в формулу для c_{max}:

$$c_{max} = \underset{c \in C}{\operatorname{argmax}} \left[P(c) \prod_{i=1}^{n} P(w_i|c) \right]$$

Теперь стоит поговорить о том, как посчитать вероятность $P(c)$ и $P(w_i|c)$. Их значение вычисляется на обучающей выборке. А вероятность определяется как:

$$P(c) = \frac{D_c}{D} \qquad\qquad P(w_i|c) = \frac{N_{ic}}{N_c}$$

где D_c – количество документов, содержащихся в классе c, D – корпус документов в обучающей выборке, N_{ic} – количество раз, сколько признак i встречается в документах класса c, N_c – общее количество признаков для класса c.

Формулу для оценки вероятности слова можно изменить - добавить сглаживание (smoothing) (эта оценка используется в полиномиальной наивной байесовской модели) [20]:

$$P(w_i|c) = \frac{N_{ic} + \alpha}{N_c + n\alpha}$$

где α - множитель сглаживания (если α = 1 метод называется сглаживанием Лапласа, а если множитель α < 1, то это сглаживание Лидстоуна), n – число признаков.

Данное сглаживание является устранением проблемы неизвестных слов – если в проверочном наборе встретилось слово, которое ранее не появлялось в тренировочном наборе данных, то оценка вероятности (без сглаживания) данного признака будет равна 0. Это значит, что документ не получится классифицировать с этим словом. А при использовании сглаживания, слова, которые не были при обучении, получают вероятность уже не равную 0.

Подставив описанные выше записи для вычисления вероятностей, получается окнчательная формула, по которой работает метод Байеса:

$$c_{max} = \underset{c \in C}{\operatorname{argmax}} \left[\frac{D_c}{D} \prod_{i=1}^{n} \frac{N_{ic}+\alpha}{N_c+n\alpha} \right]$$

Для данного метода был рассмотрен подход полиномиальной модели, которая имеет полиномиальное распределение данных, но существуют и другие варианты наивного байесовского классификатора, использующие соответствующие меры. Например, модель Бернулли [20, 21], которая реализует классификацию для данных, использующих многомерное распределение Бернулли. Отличие рассмотренного и Бернулли методов в том, что во второй – выборки должны быть представлены в виде векторов признаков с двоичным значением и при вычислении вероятности принадлежности d к c явно учитывается отсутствие признака i в классе c, а первый подход игнорирует это.

6 Описание практической части

В этой части рассмотрены программные библиотеки, которые использовались в практической части, а также описана структура файлов, содержащих программный код.

6.1 Программная реализация

Все алгоритмы реализованы на языке Python 3.9. В практической части работы использовались следующие основные библиотеки:

- **scikit-learn** – это пакет модулей, который предоставляет множество алгоритмов классификации, кластеризации и регрессии, а также возможности для предварительной обработки данных. Из данной библиотеки для решаемой задачи были взяты методы для векторизации данных и для построения классификаторов.
- **pandas** – библиотека для обработки и анализа данных. В работе используется для формирования собственных датасетов и их использования.
- **bs4** – библиотека Python для извлечения данных из файлов HTML и XML. В работе используется для изъятия текстовых данных с веб-страниц.
- **json** – библиотека, предоставляющая взаимодействие с форматом json. В работе используется для парсинга, формирования нового JSON-файла и записи в него.
- **pymorphy2** – морфологический анализатор для русского языка. В программе используется для преобразования слова в нормальную форму.
- **matplotlib** – библиотека для визуализации данных.

Код был написан в среде разработки “Jupyter Notebook”, в которой легко проверить то или иное предположение и можно работать в любой момент времени с любым фрагментом кода. И в данной работе есть несколько файлов:

- **datasets.ipunb**
- **MLMethods.ipunb**

В файле **datasets** ключевым моментом является создание своего файла в формате **.json** для хранения текстов на русском языке и некоторых других полей, связанных с обработкой веб-страниц. Для этого создан класс с закрытыми атрибутами. В котором каждый объект данного класса содержит всю основную

информацию про очередную веб-страницу: text, url и т.д. Чтобы в атрибуте “text” появились текстовые данные, производится парсинг веб-страницы. Для парсинга мною была написана функция, которая удаляет теги “script”, “style”, а затем извлекает весь текст с веб-страницы. И в конце с помощью сформировавшегося массива объектов данного класса были созданы датасеты, которые упоминались ранее, и файл в формате **.json**, содержащий объекты класса.

В файле **MLMethods** основой является обучение классификаторов на всех сформированных датасетах, для проверки различных идей, и визуализация результатов. Вначале загружается один датасет, на котором проходит обучение. И для улучшения оценки качества происходит подбор оптимальных параметров для классификаторов (LR, DT, MNB) с использованием GridSearchCV. А в качестве метрик используется “f1_micro” и площадь под кривой “полнота-точность”. После загружаются остальные для проверки методов датасеты и выводятся для них результаты обучения.

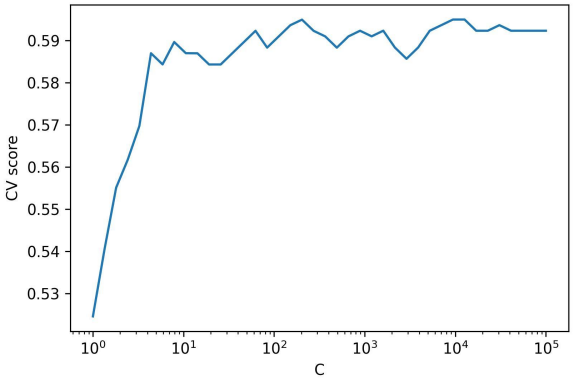

Рис. 4. Зависимость f1_score от C

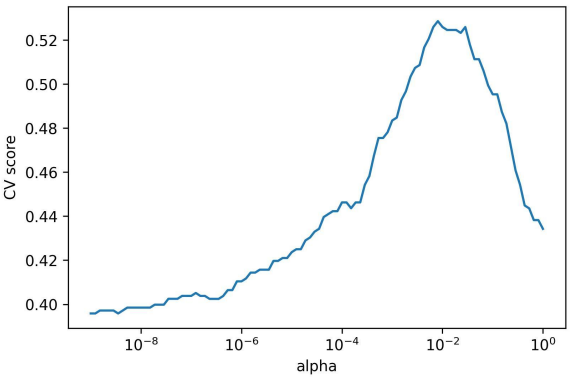

Рис. 5. Зависимость f1_score от alpha

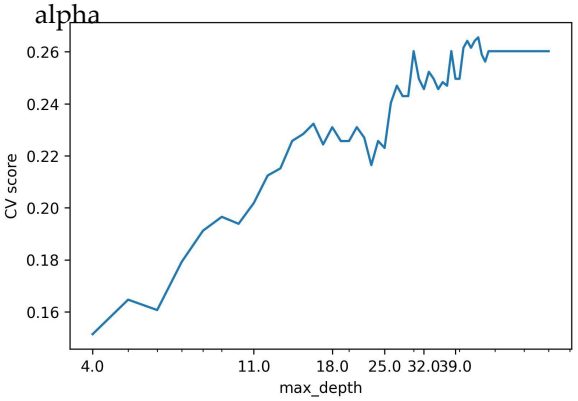

Рис. 6. Зависимость f1_score от max_depth

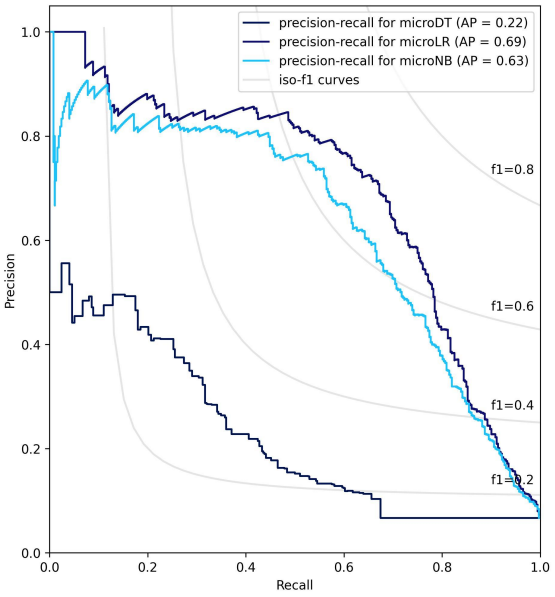

Рис. 7. PR-кривые для классификаторов

На рисунках 4-7 приведены результаты классификаторов и зависимости их параметров от оценки качества на данных из файла **dataset_75_15.csv**. (Далее во всей части 6 будут показаны результаты для этого же датасета).

6.2 Верификация алгоритмов предобработки текстов

В виду объемных текстов предоставить даже фрагменты документов до и после предобработки, кажется, не лучшей идеей. И чтобы как-то наглядно подтвердить мои алгоритмы обработки текста(включающие регулярные выражения), стоит взглянуть на рисунок 8. Для визуализации были взяты произвольные четыре текста из датасета, каждый из которых характеризуется 3 величинами:

- количество символов в начальном тексте
- количество символов после применения регулярных выражений к начальному тексту
- количество символов множества слов в нормальной форме после предыдущего этапа

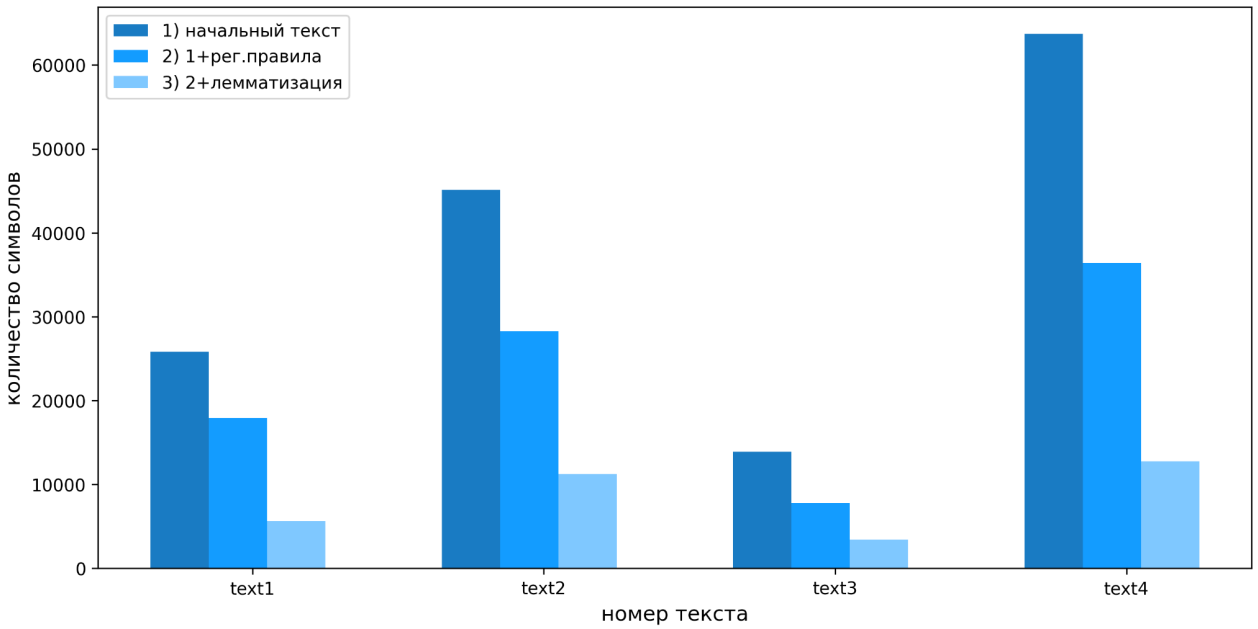

Рис. 8. Количество символов до и после предобработки для 4-х текстов

Видно, что после каждого этапа размер представленных текстов уменьшается в два раза. То есть текст, состоящий только из неповторяющихся слов в начальной форме в 4 раза меньше начального. Это связано с тем, что изначально в текстах очень много лишнего, так как они содержат буквально весь текст с веб-страницы. Например, мною было принято решение избавиться от всех слов на английском языке в текстах после парсинга, так как на веб-страницах могут содержаться

программный код или специфические слова, которые написаны обычными пользователями. Второе, для русского языка, может в некоторой степени помочь при классификации.

6.3 Подбор оптимальных параметров и верификация векторизации

Для векторизации есть несколько основных параметров, которые влияют на размер пространства признаков, а это в свою очередь на время и качество работы. Поэтому стоит посмотреть на зависимость качества моделей (logistic regression(LR), multinomial NB(MNB), decision tree(DT)) и размерности пространства признаков от параметров **min_df** и **max_df**, которые характеризуют стратегию векторизации – **Tfidf**.

min_df	Количество признаков	accuracy(DT)	accuracy(MNB)	accuracy(LR)
2	21957	0.3306	0.6236	0.6962
3	15967	0.3145	0.6182	0.6962
5	11190	0.3333	0.6263	0.6801
7	8838	0.3494	0.6182	0.6720
9	7440	0.3575	0.6129	0.6774

Таблица 1. Исследование параметра min_df

max_df	Количество признаков	accuracy(DT)	accuracy(MNB)	accuracy(LR)
0.6	15948	0.3198	0.6209	0.6881
0.3	15751	0.3252	0.6209	0.6344
0.1	14991	0.3198	0.6263	0.5887

Таблица 2. Исследование параметра max_df

Параметр **min_df** используется для удаления слов, которые появляются редко, а **max_df** – для удаления слов, которые появляются часто.

Согласно таблице 1, для высоких значений **min_df** качество классификации ухудшается. Это связано с тем, что при векторизации удаляются относительно

редкие слова, но в то же время они являются специфичными для категорий (как минимум в словаре присутствуют слова, которые писали пользователи в комментариях на той или иной веб-странице). Оптимальным выбором здесь будет **min_df = 3**, то есть слова, которые встречаются менее чем в 3 документах, будут игнорироваться.

Что касается **max_df** – этот параметр хорошо показывает себя в классификации при значении **0.6**(учитывая min_df = 3), то есть при векторизации будут игнорироваться слова, которые появляются в более чем 60% документов. Интересно, что при минимальных значениях данного параметра качество упало только у логистической регрессии, на 10%.

Теперь можно выяснить, какие слова в результате преобразования tfidf стали наиболее важными для каждой категории (таблица …)

	1	2	3	4	5	6	7	8
DATABASES	поточный	позаимствовать	корректировка	расположиться	резервирование	вышеупомянутый	ежемесячно	происхождение
DEV	онлайновый	эпизодический	сводный	десктопной	статистический	отлаживать	сериализация	предстоящий
DEVOPS	администрирование	программирование	производительность	голос	тег	перечитывать	многообразие	проза
EVENTS	доклад	ежегодный	тренинг	опубликовать	запись	проигрыватель	семинар	трансляция
HARDWARE	наметить	складываться	превращать	клава	фото	раскупить	батарея	прототип
KnD	масштабный	млмакс	долгосрочный	забагованный	минт	промежуточный	неполадка	ибилья
MISC	впечатлить	периодический	спешить	погибнуть	резкий	критика	оператор	открыться
MOBILE	мобилкаха	концепт	мобилка	оригинал	диктофон	гном	вбить	переустанавливать
MULTIMEDIA	едегор	академия	приблизиться	видеоредактор	блистать	искусство	превьюшка	ластик
ORG	биржа	акция	покупка	подняться	событие	наблюдаться	айтишник	целый
SECURITY	эксплуатироваться	брешь	псевдослучайный	хакерский	аннулировать	балл	незнание	недоумевать
SPECIAL	соответсвовать	телефония	назреть	межсетевой	systemd	дописываться	коробочный	квест
SYSADM	разворачиваться	вычисляться	сорец	выстрелить	рассказ	взятие	подводный	экзамен
USER	отклик	ибилья	приоритизация	бровь	поворот	акцент	энергопотребление	сместиться
WEB	обеспокоенность	подгружаться	затруднительно	торопить	анимационный	заплатка	усложнить	подчёркивать

Рис. 9. Слова с наибольшим tfidf для каждой категории

Из рисунка 9 видно, что слова, которые имеют значение tfidf больше 0.4, обычно являются “важными” для категории. Например, для категории “events” слова с наибольшими tfidf на самом деле описывают ее, что нельзя сказать о категории “web”: большая часть слов никак не говорит про нее (они и имеют маленький вес). И это понятно, ведь tfidf является методом неконтролируемого обучения, и слова с наибольшим tfidf не всегда бывают “важными” для категорий, то есть они необязательно должны быть связаны с интересующими рубриками.

7 Эксперименты

При исследовании качества работы алгоритмов и выбора классификатора документов были проведены эксперименты на нескольких датасетах, про которые упоминалось ранее. Далее рассмотрены результаты и на основе них сделаны выводы.

7.1 Результаты

Все ранее упомянутые классификаторы были задействованы в ходе экспериментов: логистическая регрессия, дерево решений, полиномиальный наивный байесовский классификатор. Для них изначально подобраны значения параметров. И после происходит обучение классификаторов на тестовых наборах всех датасетов. Классификатор, который имел самую высокую точность на датасетах, был **LR**. Таблица 3 показывает оценки различных классификаторов. В качестве оценки взята метрика f1 с микроусреднением, которая рассматривает весь набор данных как совокупный результат и глобально вычисляет один результат.

Метод / Обучающий датасет	**DT**	**MNB**	**LR**
dataset_150_10	0.34	0.56	**0.63**
dataset_200_15	0.31	0.52	**0.61**
dataset_75_15	0.32	0.62	**0.67**
dataset_350_4	0.61	0.77	**0.82**
dataset_200_23	0.27	0.50	**0.58**

Таблица 3. Оценки классификаторов на разных датасетах

Также отдельно рассмотрен еще один датасет **dataset_max_23.csv** – набор данных, состоящий из 23 категорий, каждая из которых содержит максимум своих записей. Этот датасет нужен для следующего эксперимента: разделить 23 категории на два класса и посмотреть, как категории с малым количеством текстов отделимы от противоположных им (здесь, конечно, не будет учитываться тематика категорий). То есть в классе “0” будут находиться 13 категорий,

имеющих больше всего записей, а в классе “1” оставшиеся категории, содержащие малое количество документов. Ввиду этого в таблице 4 показаны результаты работы классификаторов на данном датасете, где результаты подсчитаны метрикой **f1-micro**.

Поскольку логистическая регрессия имеет наибольшую оценку по сравнению с другими классификаторами, то добавлены дополнительные сведения о классификаторе: лучшие 10 признаков для нескольких категориях (**dataset_75_15,** таблица 5), матрица ошибок (рис. 10). Из таблицы 5 видно, что токены хорошо описывают свои категории.

Метод / Обучающий датасет	DT	MNB	LR
dataset_max_23	0.87	0.90	0.91

Таблица 4. Оценка бинарной классификации

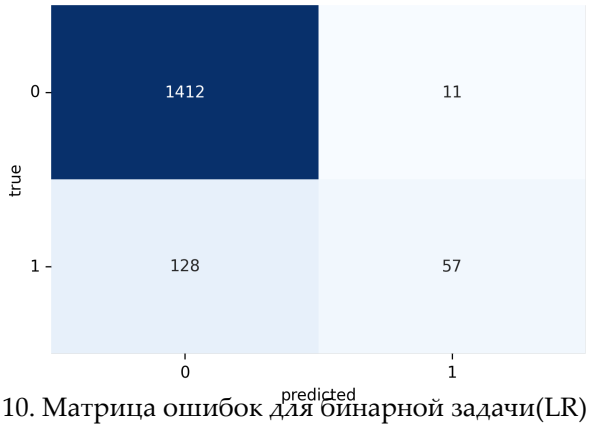

Рис. 10. Матрица ошибок для бинарной задачи(LR)

sysadm	**dev**	**multimedia**	**databases**	**security**
аналогично	среда	медиацентр	бд	атаковать
разделяться	компилировать	изображение	распределенный	устранить
пространство	указатель	звук	хранилище	затрагивать
вывести	кроссплатформенный	формат	небольшой	выявить выпустить
соединение	статический	эффект	восстановление	безопасность
флаг указывать	синтаксис	графический	база	обнаружить атака
опция	программирование	рисование	хранить	исследователь
сокет	редактор	аудио	запрос	злоумышленник
сетевой	компилятор	медиа	транзакция	
	отладчик	видео	репликация	

Таблица 5. Топ-10 слов для LR в задаче классификации для нескольких категорий

7.2 Анализ результатов

Логистическая регрессия показала хорошую оценку при использовании для классификации текста. Это можно объяснить тем, что линейные модели могут масштабироваться до очень больших наборов данных и хорошо работать с разреженными данными. Полиномиальная наивная байесовская модель также

имеет хороший результат работы и очень похожа на линейную модель (справляется с разреженными данными высокой размерности, а также работает гораздо быстрее линейной модели) [9]. И дерево решений занимают последнее место по оценке работы. Мне кажется, что дереву решений нужно несколько ключевых узлов, в то время как трудно найти "несколько ключевых токенов" для классификации текста, то есть оно не справляется в таких пространствах признаков большой размерности.

Теперь более подробно стоит поговорить об экспериментах и их результатах. Еще раз посмотрев на результаты бинарной классификации, можно заметить, что данная модель характеризуется маленьким количеством истинно положительных примеров, которых даже меньше количества ложноотрицательных. В классе “0” не все слова являются информативными (и они могут иметь высокий коэффициент), поэтому есть совпадение в токенах между классами и классификатор относит к классу “0” примеры не относящиеся к нему, а также все равно присутствует шум. Отдельно были взяты значения recall, precision для данных классов. (precision, recall, класс): (0.92, 0.99, 0), (0.84, 0.31, 1). Для класса “1” получается довольно небольшое значение полноты. Поскольку класс “0” представлен гораздо большим количеством примеров, классификатор точнее прогнозирует класс “0”. Можно попробовать увеличить значение полноты, учитывая тот факт, что случаев, когда неверно спрогнозирован класс “1” будет больше. Для этого можно снизить пороговое значение решающей функции (decision function). Удалось на немного увеличить значение полноты для класса “1” получилось 0.40.

Говоря о результатах других датасетов, можно сделать следующие выводы.

- Эксперимент с разделением на категории дал интересные для меня результаты. Предположение было, что если уменьшать количество категорий так, чтобы оставалось одинаковое количество текстов(а оно увеличивается) для каждой из них, то будет качество увеличиваться. По итогу не совсем так. Для 10 категорий, для каждой из которых дано по 150 текстов, качество ухудшилось (таблица 4). Есть вероятность, что либо какие-то категории схожи между собой, либо есть шумовые или редкие токены. Но уже для датасета с 4 категориями, содержащими каждый 350 текстов, качество заметно выросло. В принципе это объяснимо тем, что

категории не совпадают между собой, то есть разделились между собой хорошо.

- Предположение взять для 15 категорий разное количество текстов, то есть когда присутствуют мажоритарные категории, чтобы наверняка они отделились от оставшихся, не сработало. Качество лишь ухудшилось (таблица 4) в сравнении со сбалансированными данными (также 15 категорий). Для них отдельно была выведена матрица ошибок и recall, precision в практической части, чтобы посмотреть, какие классы имеют низкие значения, а также насколько много ложных срабатываний и для каких категорий. И видно, что при обучении первого датасета и второго отличились лишь некоторые классы низкой полной и точностью: "ORG", "SPECIAL", "SYSADM", "MISC".

Также интересно, что количество ложных примеров для каких-то категорий распределяется на конкретные классы. Например, для класса "KnD" с возрастанием количества текстов, возрастает и количество ложных примеров как класс "USER". Исходя из этого примера, в бинарной классификации эти категории содержатся в классе "0" и из-за этого уменьшается количество ложных примеров для класса "0". Хотя если смотреть на слова с наибольшими коэффициентами для каждой из категорий, то нельзя сказать, что они как-то схожи между собой ("KnD": свежий, подсистема устанавливаться, переустановка, накопитель, целостность, редакция, бета, основать, драйвер; "USER": понравиться, флешка, копировать, окно, офисный, меню, мышь, находить, оболочка, стол).

Теперь сравним полученные результаты с результатами других аналогичных решений (все сравниваемые результаты показаны в таблице 6). В качестве критерия будет использоваться метрика f1-мера.

В первую очередь рассмотрим результаты решения [5]. Целью данной работы также классифицировать веб-страницы, то есть сравнение будет с наиболее похожим решением. На вход подается 8 категорий, каждая из которых содержит 150 текстов. Категории у автора не пересекаются друг с другом (например, medical и politics), поэтому алгоритмы могут лучше справляться с разделением. Результаты автора: LR – 0.925, DT – 0.758, MNB – 0.917 против моих: LR – 0.63, DT – 0.34, MNB – 0.56 (результаты – таблица 3, датасет с 10 категориями по 150 текстов).

Результаты сильно отличимы. Заметим, что по убыванию классификаторы у автора расположены также, то есть логистическая регрессия также работает лучше остальных методов. Автор приводит в текстовом описании, с каких веб-страниц берутся текстовые данные, и есть предположение, что ссылки не содержат комментарии пользователей, из-за этого шума должно меньше.

В следующей работе [22] также автор классифицирует веб-страницы. На входе 8 категорий. Данные несбалансированные (например, в одной категории 17 текстов, в другой – 789). Автор с веб-страниц изымает разные теги-заголовки и по каждому из них делает свой датасет. На выходе следующие результаты для MNB: 0.568. В моем случае для 10 категорий классификатор MNB имеет результат 0.56, что почти также. Качество работы классификаторов на текстовых данных в разных заголовках очень отличается у автора.

В другом решении [2] начальные данные содержат 20 новостных категорий с 1000 текстами на каждую категорию. Здесь задействованы другие методы машинного обучения – метод монотонного ближайшего соседа monNN, и результат для него 0.663 (f1-macro). Если брать в сравнение мой датасет с 15 категориями, то LR имеет результат 0.67, а если датасет с 23 категориями, то 0.58. Если быть более честным, то можно посмотреть на результат работы логистической регрессии конкретно для этого датасета в работе [23], тут он 0.596 (приблизительно такой же, как у первого автора с другим классификатором).

Поскольку сравниваемые результаты сильно обобщены и все зависит от применения решения к конкретной реальной задачи, то явных преимуществ нет у рассмотренных решений. Но учитывая все это, логистическая регрессия для данной задачи показала себя хорошо.

Метод / Обучающий датасет	DT	MNB	LR
результаты курсовой	0.34	0.56	0.63
работа [5]			
работа [22]		0.56	
работа [23]			0.6

Таблица 6. Сравниваемые результаты

8 Заключение

В рамках курсовой работы решалась задача категоризации текстов с помощью методов машинного обучения. И основные результаты работы:

1. Изучены методы машинного обучения для решения задачи категоризации текстов;
2. Разработаны и реализованы методы сбора текстовых данных с веб-страниц и методы предварительной обработки текстов;
3. Реализованы алгоритмы, основанные на методах Logistic Regression, Decision Tree, Multinomial Naive Bayes;
4. Проведен сравнительный анализ методов машинного обучения. По полученным результатам выполнено сравнение с результатами других аналогичных решений. И выбран наиболее подходящий метод, на основе которого будет разработана система категоризации веб-страниц.

На данном этапе есть также ряд недостатков:

1. В данной курсовой работе используется упрощенный подход к выбору признаков: исключаются стоп-слова и работает лемматизация. Вполне вероятно, что скорость и точность классификации могут увеличиться, если будут использоваться более продвинутые методы выбора признаков. Извлеченный текст может быть зашумлен из-за характера текстового контента в интернете. Поэтому от такого надо тщательнее избавляться.
2. По получившимся результатам пока нельзя использовать данные методы классификации текста для категоризации веб-страниц в реальных задачах. Хотя результаты могут быть связаны с ошибками в данных, а не в подходе. Поэтому нельзя не задаться вопросом, могут ли быть подходящими другие методы.

Перспективы работы над темой:

1. Рассмотреть различные методы отбора признаков и экспериментально проверить на эффективность.
2. Реализовать систему категоризации веб-страниц с графическим интерфейсом, которая включает классификацию с веб-страницами на английском языке.
3. Рассмотрение моделей для получения эмбеддингов и включение их в свой алгоритм классификации веб-страниц. Возможно, применение языковой модели BERT.

Литература

[1] Автоматическая обработка текстов на естественном языке и анализ данных: учеб. пособие / Большакова Е.И., Воронцов К.В., Ефремова Н.Э., Клышинский Э.С., Лукашевич Н.В., Сапин А.С. 2017.

[2] Романенко А.А. Категоризация текстов на основе монотонного классификатора ближайшего соседа // Дипломная работа, 2012. URL: http://www.machinelearning.ru/wiki/images/0/02/Romanenko2012bach.pdf

[3] Сравнительный анализ методов машинного обучения для решения задачи классификации документов научно-образовательного учреждения / М. Н. Краснянский, А. Д. Обухов, Е. М. Соломатина, А. А. Воякина. // Вестник ВГУ, серия: системный анализ и информационные технологии, 2018, No 3. URL: http://www.vestnik.vsu.ru/pdf/analiz/2018/03/2018-03-19.pdf

[4] Thorsten Joachims, Text Categorization with Support Vector Machines: Learning with Many Features // Proceedings of the 10th European Conference on Machine Learning, 1998, URL: https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf

[5] Patrick Dave P. Woogue, Gabriel Andrew A. Pineda. Automatic Web Page Categorization Using Machine Learning and Educational-Based Corpus // IJCTE, Vol. 9, No. 6, 2017, URL: http://www.ijcte.org/vol9/1180-IT026.pdf

[6] Clarence Chio and David Freeman. Machine Learning and Security: Protecting Systems with Data and Algorithms. 2018.

[7] Daniel Jurafsky & James H. Martin. Regular Expressions, Text Normalization, Edit Distance // Speech and Language Processing book, chapter 2, URL: https://web.stanford.edu/~jurafsky/slp3/2.pdf(дата обращения: 20.02.2022)

[8] Vimala Balakrishnan and Ethel Lloyd-Yemoh. Stemming and Lemmatization: A Comparison of Retrieval Performances // Lecture Notes on Software Engineering, Vol. 2, No. 3, 2014, URL: http://www.lnse.org/papers/134-I3007.pdf(дата обращения: 20.02.2022)

[9] Andreas Mueller. Introduction to Machine Learning with Python: A Guide for Data Scientists. 2017.

[10] Документация библиотеки scikit-learn [Электронный ресурс]. // Метод уменьшения размерности - RFE, URL: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE(дата обращения: 20.03.2022)

[11] Документация библиотеки mlxtend [Электронный ресурс]. Метод уменьшения размерности – SFS. – URL: http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/(дата обращения: 20.03.2022)

[12] Т.В.Батура. Методы автоматической классификации текстов. // Международный журнал “Программные продукты и системы”, 2016, URL: https://www.researchgate.net/publication/315328102_Metody_avtomaticeskoj_klassifikacii_tekstov(дата обращения: 20.03.2022)

[13] Мотренко А.П., Стрижов В.В. Многоклассовая логистическая регрессия для прогноза вероятности наступления инфаркта. URL: http://strijov.com/papers/Strijov-Motrenko2012HAPrediction_Preprint.pdf(дата обращения: 27.03.2022)

[14] Logistic Regression: Loss and Regularization [Электронный ресурс]. https://developers.google.com/machine-learning/crash-course/logistic-regression/model-training(дата обращения: 27.03.2022)

[15] Chris Elbon. Machine learning using Python. Recipe Book. 2018.

[16] Harsh H. Patel, Purvi Prajapati. Study and Analysis of Decision Tree Based Classification Algorithms. // JCSE. – 2018. Vol.-6, Issue-10. URL: https://www.researchgate.net/publication/330138092_Study_and_Analysis_of_Decision_Tree_Based_Classification_Algorithms(дата обращения: 2.04.2022)

[17] А. Г. Дьяконов. Индекс Джини [Электронный ресурс]. URL: https://dyakonov.org/2015/12/15/знакомьтесь-джини/(дата обращения: 2.04.2022)

[18] Е. А. Соколов. Решающие деревья: Лекция 3 // – 2018. URL: https://www.hse.ru/mirror/pubs/share/215285956(дата обращения: 3.04.2022)

[19] А.Г. Дьяконов. Байесовский подход [Электронный ресурс]. URL: https://dyakonov.org/2018/07/30/байесовский-подход/(дата обращения: 5.04.2022)

[20] Документация библиотеки scikit-learn. Наивный метод Байеса [Электронный ресурс]. URL: https://scikit-learn.ru/1-9-naive-bayes/(дата обращения: 5.04.2022)

[21] Александр Сизов, Сергей Николенко. Наивный байесовский классификатор: Презентация. URL: https://www.dropbox.com/s/9r7pte8ybtj6khc/Ссылка%20на%20sem01-naivebayes.pdf?dl=0(дата обращения: 7.04.2022)

[22] https://scikit-learn.org/stable/auto_examples/linear_model/plot_sparse_logistic_regression_20newsgroups.html
https://www.diva-portal.org/smash/get/diva2:700316/FULLTEXT01.pdf

[23] https://scikit-learn.org/stable/auto_examples/linear_model/plot_sparse_logistic_regression_20newsgroups.html