Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 01.03.02 Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент, доцент кафедры УМФ,
к.ф.-м.н.
_____/ Е.В. Бычков
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
профессор
_____/А.А.Замышляева
«____»_____20___ г.

Построение признакового пространства для визуальных элементов web-страниц

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2020.072.ПЗ ВКР

Руководитель работы,
к.ф.-м.н., доцент
_____/С.М. Елсаков
«____»_____2020 г.
Автор работы
Студент группы ЕТ-412
_____/ Д.С. Ткачев
«____»_____2020 г.
Нормоконтролер,
ст. преподаватель
_____/Н.С. Мидоночева
«____»_____2020 г.

Челябинск
2020

АННОТАЦИЯ

Ткачев Д.С. Построение признакового пространства для элементов веб-страниц. – Челябинск: ЮУрГУ, ЕТ-412, 51 с., 26 ил., 3 табл., библиогр. список – 40 наим., 1 прил.

Данная работа посвящена построению признакового пространства для элементов веб-страниц, позволяющего создать удобный инструмент для сбора данных с веб-страниц. Отличием разработанного программного обеспечения является удобство, простота эксплуатации и возможность его использования рядовым пользователем, не имеющего профессиональных знаний и умений в области веб-технологий.

Целью данной работы является построение признакового пространства для элементов веб-страниц и его тестирование с помощью программной реализации.

В первой главе рассмотрены методы поиска аналогов и анализа признакового пространства, приведено обоснование необходимости в разработке программного обеспечения.

Во второй главе рассмотрены методы и алгоритмы решения данной задачи, разработана архитектура приложения для проверки их работоспособности.

В третьей главе проведена разработка расширения для веб-браузера и его тестирование. Описана архитектура приложения, инструментов разработки и получены результаты тестов на выборке.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 7
1 МЕТОДЫ, АЛГОРИТМЫ И МОДЕЛИ АНАЛИЗА ВЕБ-СТРАНИЦ 9
1.1 Методы машинного обучения 9
1.1.1 Деревья решений 10
1.1.2 Наивный байесовский классификатор 13
1.1.3 Выбор общих параметров 15
1.2 Методы анализа признакового пространства 16
1.2.1 Корреляция 16
1.2.2 Автокорреляция 18
1.2.3 Факторный анализ 20
1.3 Подготовка данных для анализа 22
1.4 Описание предметной области 22
1.4.1 Веб-скрейпинг 22
1.4.2 Объектная модель документа 24
2 ПОСТРОЕНИЕ ПРИЗНАКОВОГО ПРОСТРАНСТВА 27
2.1 Постановка задачи 27
2.2 Формирование обучающей выборки 27
2.3 Пример признакового пространства для одной из веб-страниц 28
2.4 Выбор метода для поиска аналога 30
2.4.1 Деревья решений 30
2.4.2 Наивный байесовский классификатор 31
2.4.3 Выбор общих параметров 32
2.5 Отбор оптимальных признаков 33

2.6 Выводы по разделу 37
3 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ТЕСТИРОВАНИЯ МЕТОДА ПОИСКА АНАЛОГОВ 38
3.1 Разработка архитектуры приложения 38
3.2 Проверка корректности работы 41
3.4 Выводы по разделу 46
ЗАКЛЮЧЕНИЕ 47
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 48
ПРИЛОЖЕНИЕ 1 Исходный код 52

ВВЕДЕНИЕ

В современном мире, нас окружает огромное количество информации, которая, чаще всего, хранится в электронном виде на веб-сайтах. У каждого магазина, новостного агентства или другого сервиса есть сайт, с которого можно получить какие-то данные, например, список доступных товаров или актуальных новостей. Люди используют социальные сети, где публикуют свои личные данные: имя, фамилию и номер телефона.

Образуется конкуренция в различных сферах деятельности людей, например, в бизнесе и рекламе. Она требует активного сбора данных и делает крайне востребованными подходы по их оперативному получению из уже имеющейся открытой информации.

Такой большой поток не отсортированных сведений может показаться нам бесполезным, но в нем содержится ценная информация, которую можно использовать в бизнесе, рекламе и других сферах деятельности.

В ходе стремительного развития веб-технологий появляются новые способы для улучшения взаимодействия каждого пользователя с интернет страницами. Интернет полон открытых данных, но требует определенных специальных средств для их сбора.

Из-за популярности этой сферы, была сформирована область программной инфраструктуры, поддерживающей накопление и постоянное пополнение архивов данных различной природы и назначения.

Проблема обеспечения простого и быстрого сбора больших объемов данных из сети стала более актуальной. Возникает потребность в определенном классе программных средств, которые станут удобным и простым в использовании инструментом для рядового пользователя, не имеющего продвинутых знаний в области современных веб-технологиях, при этом имеющего высокую скорость сбора корректных данных на различных веб-страницах.

Для комплексного выполнения таких требований нужно использовать новые подходы и методы в разработке, способные решить все основные проблемы создания и работы систем данного класса.

Цель данной работы – построение признакового пространства для веб-страниц и их визуальных элементов.

Задачи данной работы:

– выполнить обзор методов машинного обучения, предназначенных для решения задачи классификации;

– описать оптимальное признаковое пространство для веб-страниц и их элементов;

– выполнить программную реализацию приложения, которое извлекает структурированные данные с веб-страниц;

– провести тестирование разработанного приложения.

1 МЕТОДЫ, АЛГОРИТМЫ И МОДЕЛИ АНАЛИЗА ВЕБ-СТРАНИЦ

1.1 Методы машинного обучения

Главной задачей построения признакового пространства для элементов веб-страниц является его применение в программном обеспечении для сбора данных из открытых источников. Страницы будут выбираться пользователем, поэтому признаковое пространство должно быть универсальным для большинства веб-страниц.

Машинное обучение является ветвью искусственного интеллекта, основной идей которого, является работа по заранее написанному алгоритму, а также самостоятельное обучение решению поставленной задачи [4]. Это процесс, в ходе которого система обрабатывает множество примеров, выявляет закономерность и использует их в дальнейшей работе, для прогнозирования характеристик новых данных.

Машинное обучение подразделяется на два направления обучения: с учителем и без. В данном случае «учителем» является вмешательство человека в процесс обработки данных.

Обучение без учителя подразумевает под собой использование только тех данных, свойств и закономерностей, которые необходимо найти. Тренировочный набор не имеет размеченных данных, они не относятся ни к какому классу (кластеру).

У обучения с учителем ситуация отличается. Тренировочный набор состоит из маркированных данных. Целью данного направления является нахождение на определенном тестовом наборе данных схожих свойств объектов, которые возможно будет применить на другом наборе данных с такими же результатами распознавания [3]. Если у множества возможных ответов имеется ограниченное количество значений, то возникает задача классификации.

На выбор метода машинного обучения влияет множество различных факторов, таких как структура, объем и качество набора данных и необхо-

димый результат [1]. Для достижения поставленной цели необходимо выбрать подходящий метод машинного обучения на основе исходных данных.

1.1.1 Деревья решений

Деревья решений (decision trees) – это один из наиболее популярных подходов для Data Mining задач. Они создают древовидную иерархическую структуру классифицирующих правил «если-то». Для принятия решения, к какому классу следует отнести объект, требуется ответить на вопросы, находящиеся в узлах дерева, начиная с его корня (рисунок 1.1).

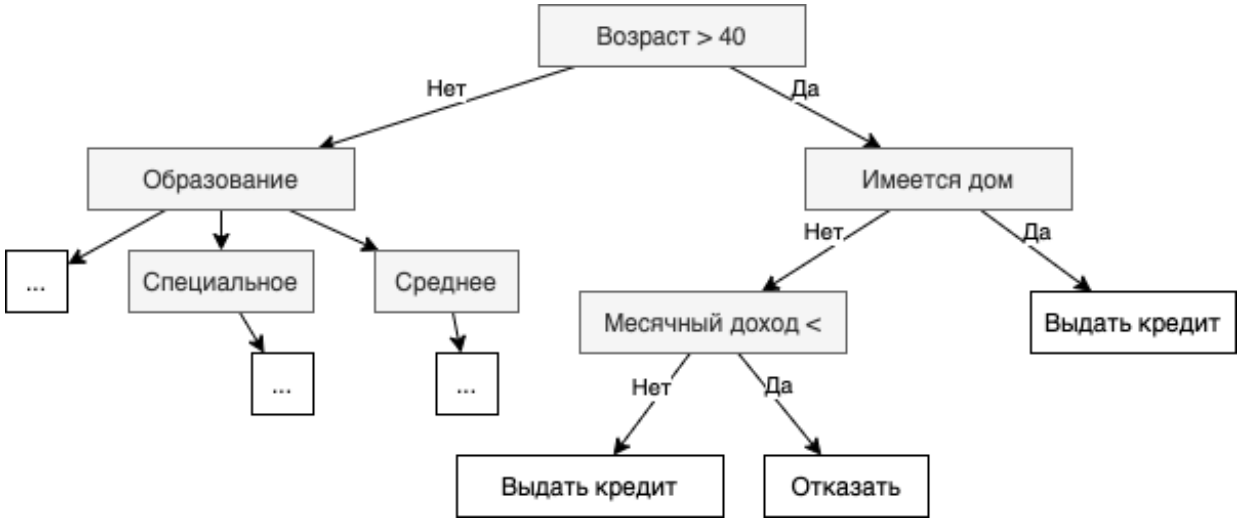

Рисунок 1.1 – Схема дерева решений

Каждый вопрос имеет ответ «да» или «нет». После каждого ответа осуществляется переход на следующий уровень иерархии и снова задается вопрос. Такие деревья отлично справляются с задачами классификации, т.е. позволяют отнести объект к заранее известному классу.

Дерево строится от корня, при этом для него выбирается атрибут для проверки. Каждый атрибут исследуется на предмет того, насколько хорошо он в одиночку классифицирует набор данных. Когда такой атрибут выбран, для каждого его значения создается ветка дерева, а набор данных делится в соответствии со значением в каждой ветке, при этом процесс выполняется рекурсивно для каждой ветки.

Для построения дерева решений готовится обучающая выборка. Выходное поле может быть только одно и только дискретного типа. Стремление учесть как можно больше потенциально полезной информации приводит к включению избыточных шумовых признаков. Для успешного обучения число примеров должно в несколько раз превосходить число входных признаков. Если данных мало, приходится упрощать структуру регрессионной модели, оставляя только наиболее существенные признаки.

После того, как определились с входными и выходными полями, следует нормализация данных в обучающей выборке. Ее цель – это преобразование данных к виду, наиболее подходящему для обработки алгоритма.

Деревья решений помогают оценить значения категориальных и количественных атрибутов (конечное число дискретных значений). В первом случае говорят о задаче классификации (отнесение объекта к одному из описанных классов). Второй случай – это задача о регрессии, то есть об оценке количественной величины.

Задачей оценочной функции является уменьшение неопределённости в узле. Идея неопределённости формализована в индексе Gini. Если набор данных T содержит данные n классов, тогда индекс Gini определяется формулой (1.1).

$$g(T) = 1 - \sum_{i=1}^{n} p_i^2, \tag{0.1}$$

где p_i – вероятность (относительная частота) класса i в T.

Если набор T разбивается на две части T_1 и T_2 с числом примеров в каждом N_1 и N_2 соответственно, тогда показатель качества разбиения вычисляется по формуле (1.2).

$$gs(T) = \frac{N_1}{N} g(T_1) + \frac{N_2}{N} g(T_2), \tag{0.2}$$

где N – число примеров в узле-предке.

Наилучшим считается то разбиение, для которого $gs(T)$ минимально.

Если обозначить L, R – число примеров соответственно в левом и правом потомке, l_i и r_i – число экземпляров i-го класса в левом и правом потомке. Тогда качество разбиения оценивается по формуле (1.3).

$$gs = \frac{L}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{r_i}{R}\right)^2\right) \to min. \quad (0.3)$$

Чтобы уменьшить объем вычислений формулу (1.3) можно преобразовать, как показано в формуле (1.4).

$$gs = \frac{1}{N}\left(L\left(1 - \frac{1}{L^2}\sum_{i=1}^{n} {l_i}^2\right) + R\left(1 - \frac{1}{R^2}\sum_{i=1}^{n} {r_i}^2\right)\right) \to min. \quad (0.4)$$

Умножение на константу не влияет на минимизацию, формулу (1.4) можно переписать, как показано в формулах с (1.5) по (1.7).

$$gs = L - \frac{1}{L}\sum_{i=1}^{n} {l_i}^2 + R - \frac{1}{R}\sum_{i=1}^{n} {r_i}^2 \to min, \quad (0.5)$$

$$gs = N - \left(\frac{1}{L}\sum_{i=1}^{n} {l_i}^2 + \frac{1}{R}\sum_{i=1}^{n} {r_i}^2\right) \to min, \quad (0.6)$$

$$\widetilde{gs} = \frac{1}{L}\sum_{i=1}^{n} {l_i}^2 + \frac{1}{R}\sum_{i=1}^{n} {r_i}^2 \to max. \quad (0.7)$$

В итоге, лучшим будет то разбиение, для которого величина максимальна.

Преимущества алгоритма:

– наглядность и простота интерпретации;

– универсальность в решении задач классификации и регрессии;

– возможность работы с категориями и с количественными значениями.

Недостатки:

– сложность и размера дерева. Размер дерева является критическим фактором, определяющим качество решения задачи;

– нестабильность процесса. Даже небольшие изменения в наборе данных могут привести к построению нового дерева. Это связано с

иерархичностью дерева решений. Изменения в узле на верхнем уровне ведут к изменениям во всем дереве ниже;

– является вычислительно затратной задачей и имеет вычислительную сложность порядка $O(2^n)$, где n – количество принимаемых признаком значений.

1.1.2 Наивный байесовский классификатор

Наивный байесовский классификатор (naive Bayes) – это частный случай байесовского классификатора, который основан на дополнительном предположении, что объекты описываются статистически независимыми признаками.

Например, имеется набор данных о человеке:

– вес;

– рост;

– температура;

– номер телефона.

Такие параметры будут независимыми, если их значения не оказывают влияния друг на друга. Очевидно, что вес человека никак не зависит от его номера телефона.

Метод назван наивным, так как он предполагает, что все параметры определенного набора данных являются независимыми, что представляет собой наивное предположение [5, 34]. Класс предсказывается на основе набора параметров, используя вероятность. Байесовские оценки вероятностей могут сильно отличаться от реальных, так как на практике чаще всего признаки являются зависимыми. Несмотря на это, классификатор имеет хорошие результаты, которые редко удается существенно превзойти.

Наивные байесовский классификатор чаще всего используется как примитивный эталон для сравнения различных моделей алгоритмов, либо как строительный блок в алгоритмических композициях [12].

Постоянство количества признаков является существенным условием при выявления взаимосвязи между ними, так как в основе признаков находится байесовская оценка [18]. Вероятностная модель классификатора – условная модель, представленная в формуле (1.8) над независимой переменной класса C и признаками $x_1, x_2, \dots, x_n$ по теореме Байеса:

$$p(C|x_1, x_2, \dots, x_n) = \frac{p(C)p(x_1, x_2, \dots, x_n|C)}{p(x_1, x_2, \dots, x_n)}, \tag{0.8}$$

где $p(C|x_1, x_2, \dots, x_n)$ – апостериорная вероятность;

$p(C)$ – априорная вероятность;

$p(x_1, x_2, \dots, x_n|C)$ – условная вероятность;

$p(x_1, x_2, \dots, x_n)$ – полная вероятность;

n – количество признаков.

В свою очередь знаменатель представляет собой множитель, зависящий только от признаков $x_1, x_2, \dots, x_n$. Числитель, представленный в формуле (1.9), эквивалентен совместной вероятности модели.

$$p(C)p(x_1, \dots, x_n|C) = p(C)p(x_1|C) \dots p(x_n|C, x_1, \dots, x_{n-1}). \tag{0.9}$$

Далее в байесовском классификаторе используется предположение о независимости признаков $x_1, x_2, \dots, x_n$, представленном в формуле (1.10).

$$p(C)p(x_1, x_2, \dots, x_n|C) = p(C)\prod_{i=1}^{n} p(x_i|C). \tag{0.10}$$

Основные преимущества этого классификатора:

– простота реализации;

– низкие вычислительные затраты при обучении и классификации;

– оптимален, когда признаки действительно независимы.

Недостатками являются:

– относительно низкое качество в большинстве реальных задач.

1.1.3 Выбор общих параметров

Еще одним методом стал выбор общих параметров. Такой метод очень прост для реализации и имеет высокую скорость работы. Главной проблемой в этом методе станет выбор начальных признаков объекта.

Выбирается набор признаков, которые могут повлиять на выбор объекта из множества. После создается обучающая выборка из несколько элементов, при этом выбираются те параметры, которые совпадают во всех объектах выборки (рисунок 1.2).

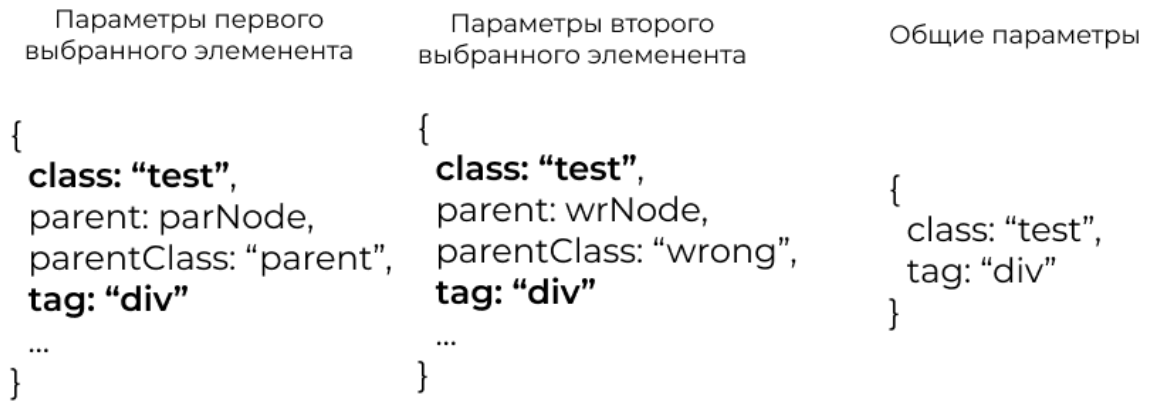

Рисунок 1.2 – Выбор общих параметров

Другими словами, для первого и второго выбранного элемента создается множество признаков A и B соответственно. Общим множеством становится их пересечение $A \cap B = \{x | x \in A \wedge x \in B\}$, которое состоит из всех элементов, принадлежащих одновременно всем данным множествам (рисунок 1.3).

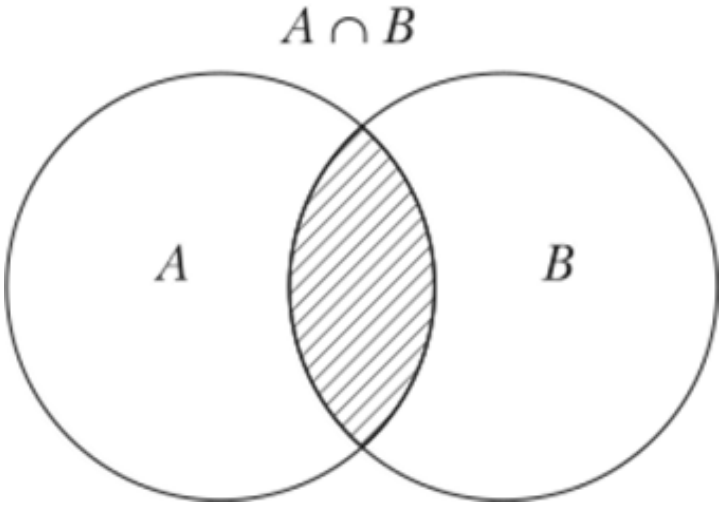

Рисунок 1.3 – Пересечение множеств

Преимуществом данного метода является его быстродействие и простота.

Недостатком – алгоритм работает не всегда точно, потому что большую роль играет множество параметров выбранных в начале.

1.2 Методы анализа признакового пространства

1.2.1 Корреляция

Статистическую взаимосвязь двух или нескольких случайных величин, либо степень связи между двумя переменными называют корреляцией. Изменение одной из таких величин может привести к систематическому изменению других. Математической мерой корреляции двух случайных величин называют коэффициентом корреляции [6, 7].

Возьмем две переменные x и y и предположим, что они являются количественными, например, рост и вес человека. Измерим пару величин (x, y) у каждого из n объектов в выборке. Для каждой соответствующей пары величин можно нарисовать точку на двумерном графике рассеяния точек.

Расположив на графике переменную x по горизонтальной оси, а y – на вертикальной, поместив пару точек для каждого объекта, получим график рассеяния точек, который показывает соотношение между двумя этими переменными.

Соотношение переменных считают линейным, если проведенная прямая линия через центральную часть скопления точек покажет наиболее подходящую аппроксимацию наблюдаемого соотношения. Для измерения близости к прямой линии вычисляется коэффициент корреляции Пирсона, который всегда варьируется от -1 до 1.

Некоторые виды корреляционных связей распределяются на положительные и отрицательные. В первом случае, увеличение одной переменной связано с уменьшением другой. В другом, при возрастании одной переменной вызывается увеличение другой (рисунок 1.4).

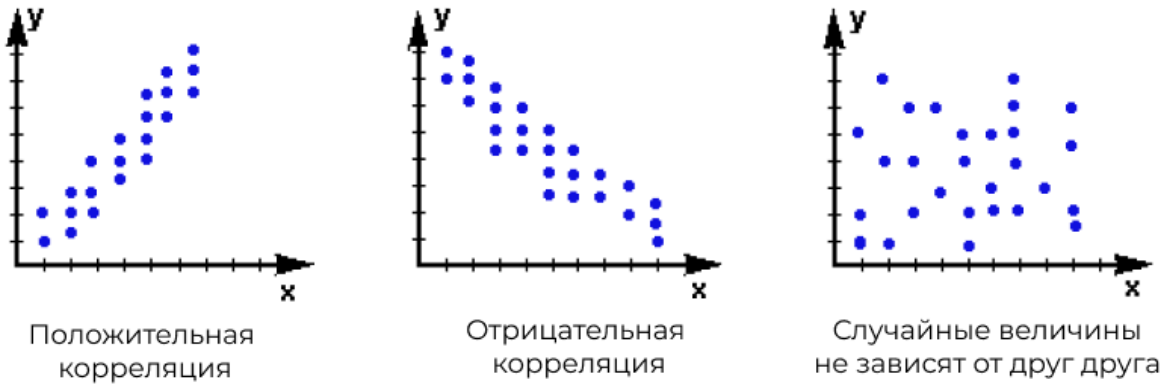

Рисунок 1.4 – Определение корреляции по диаграмме

Пусть $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ – выборка из n наблюдений пары переменных (X, Y), тогда выборочный коэффициент корреляции Пирсона r определяется как:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{0.11}$$

где $\bar{x}, \bar{y}$ – выборочные средние, определяющиеся следующим образом:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{0.12}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{0.13}$$

При этом необходимо выполнение следующих условий:

– исследуемые переменные должны быть распределены нормально;

– исследуемые переменные должны быть измерены в интервальной шкале или шкале отношений;

– количество значений в исследуемых переменных X и Y должно быть одинаковым.

Слабыми сторонами линейного коэффициента корреляции Пирсона является возможность его работы только с линейной взаимосвязью. Такая модель обычно оказывается бесполезной, если поведение выходной переменной никак не связано с входной. При этом наличие корреляции может

вызвать серьезные проблемы, связанные с неустойчивостью модели, излишней размерностью, сложностью и низкой точностью.

1.2.2 Автокорреляция

Мера корреляции между функцией и ее копией, сдвинутой на некоторый интервал (лаг), в математической статистике называют автокорреляцией. Она используется при анализе временного ряда в целях обнаружения каких-либо закономерностей в поведении ряда, а также построения прогноза для дальнейшего развития.

Для автокорреляционного анализа рассчитываются коэффициенты корреляции (мера взаимной зависимости) для двух значений выборки, находящиеся друг от друга на расстоянии лага. Количественно автокорреляцию можно измерить с помощью коэффициента автокорреляции.

Для временного ряда $X_1, X_2, \dots, X_{n-L}$ и его копии $X_{1+L}, X_{2+L}, \dots, X_n$, сдвинутой на лаг L, коэффициент автокорреляции высчитывается следующим образом:

$$r_{t,t-L} = \frac{\overline{X_t X_{t-L}} - \bar{X}_t \bar{X}_{t-L}}{s_t s_{\,t-L}}, \tag{0.14}$$

где $\overline{X_t X_{t-L}} = (\sum_{t+L}^{n} X_t X_{t-L})/(n-L)$;

$\overline{X_t} = (\sum_{t+L}^{n} X_t)/(n-L)$ – среднее значение исходного ряда;

$\overline{X_{t-L}} = (\sum_{t+L}^{n} X_{t-L})/(n-L)$ – среднее значение сдвинутого ряда;

s_t – среднеквадратическое отклонение исходного ряда;

s_{t-L} – среднеквадратическое отклонение сдвинутого ряда.

Временной лаг определяет порядок коэффициента автокорреляции. Например, если $L = 1$, то получим коэффициент автокорреляции 1-го порядка r_{t-L}. Рекомендуют максимальный порядок коэффициента автокорреляции равный $n/4$, так как с увеличением лага на единицу число пар значений уменьшается на 1.

Можно рассчитать несколько коэффициентов автокорреляции и определить лаг L' при котором автокорреляция наиболее высокая, определив

тем самым структуру временного ряда. Исследуемый ряд содержит только тенденцию, а циклическая составляющая отсутствует, если наиболее высоким значением оказывается $r_{t,t-L'}$.

В другом случае, если наиболее высоким оказался r_{t-L}, то ряд, содержит и тенденцию и циклические колебания с периодом L.

Автокорреляция наблюдается тогда, когда значения предыдущих лагов завышают или занижают значения последующих. Положительная автокорреляция на графике имеет тенденцию сохранять направление от наблюдения к наблюдению, а отрицательная предполагает наличие тенденции к постоянной смене направления (рисунок 1.5).

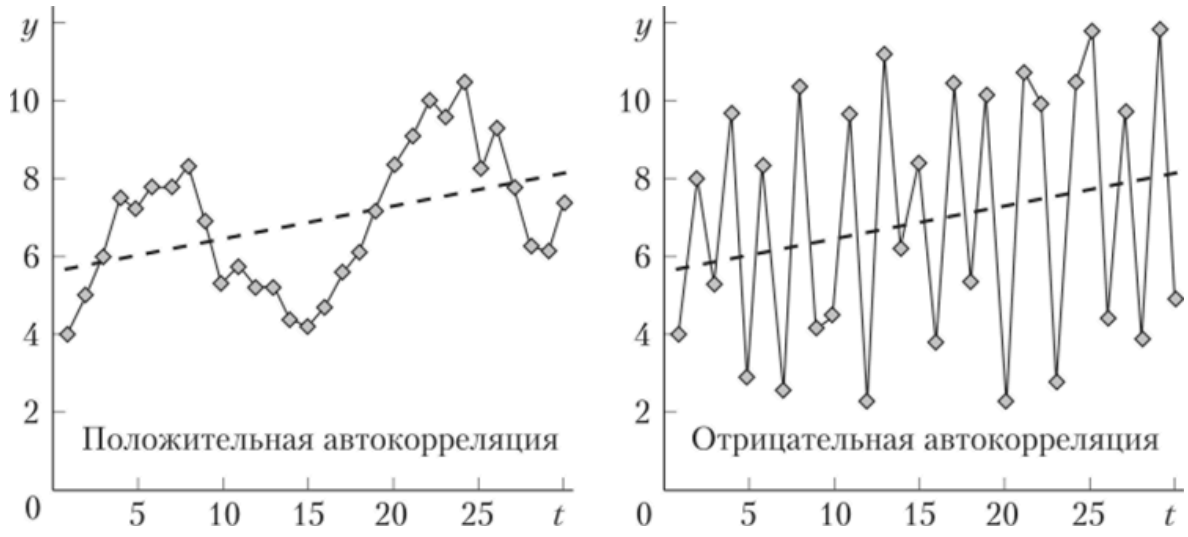

Рисунок 1.5 – Определение корреляции по диаграмме

Автокорреляционной функцией $f(L)$ называют последовательность коэффициентов автокорреляции. Она достигает максимального значения, равное 1 при $L = 0$, что говорит нам о том, что ряд полностью коррелирован сам собой, либо равное 0, когда исходный ряд и его сдвинутая копия не коррелированы.

Чем быстрее убывает автокорреляционная функция с ростом L, тем слабее автокорреляция, и наоборот.

1.2.3 Факторный анализ

Часто, в наборе данных объекты описываются большим числом признаков, из-за чего могут возникнуть проблемы наличия слабоинформативных или неинформативных признаков.

Факторный анализ – это многомерный статистический метод, который применяется для изучения взаимосвязей между значениями количественных переменных. Основная идея факторного анализа заключается в том, что имеющиеся зависимости между большим числом исходных наблюдаемых переменных определяются существованием гораздо меньшего числа переменных, называемых факторами.

Анализ большого числа признаков на наличие неинформативности крайне затруднительно, поэтому будет уместно применить метод главных компонент, суть которого состоит в замене коррелированных компонентов некоррелированными факторами.

Важной характеристикой данного метода является возможность ограничить наиболее информативные главные компоненты и исключить другие из анализа. Это намного упрощает интерпретацию результатов.

Метод главных компонент является единственным математически обоснованный метод факторного анализа. Он представляет собой ортогональное линейное преобразование, которое отображает данные из исходного пространства признаков в новое пространство меньшей размерности.

Первая ось (первая главная компонента) новой системы строится так, чтобы дисперсия данных вдоль нее была бы максимальна. Вторая ось строится ортогонально первой, при этом дисперсия для нее должна быть максимальной из оставшихся возможных (рисунок 1.6).

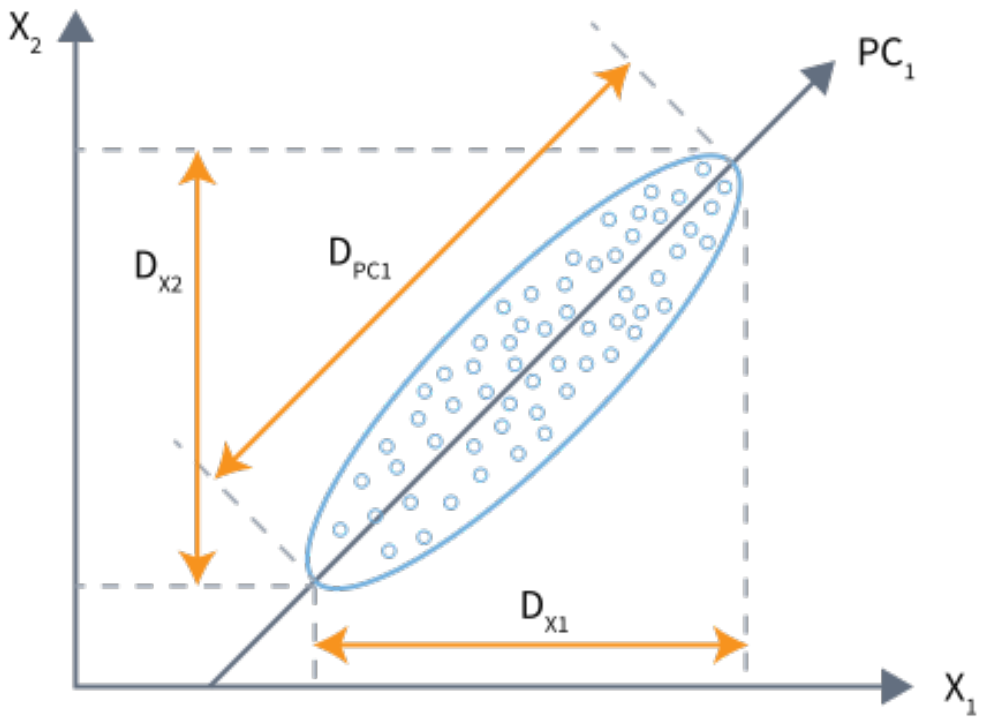

Рисунок 1.6 – Снижение размерности пространства

На рисунке 1.6 показано снижение размерности исходного двумерного пространства (X_1, X_2) с помощью метода главных компонент до одномерного. Первая главная компонента PC_1 проходит вдоль направления наибольшей вытянутости эллипсоида рассеяния точек объектов исходного набора данных в пространстве признаков, поэтому с ней связана наибольшая дисперсия.

Также, можно увидеть, что проекция дисперсии данных на ось 1-ой главной компоненты D_{PC_1}, больше, чем ее проекция на исходные оси D_{X_1} и D_{X_2}, но меньше их суммы. Из этого видно, что с первой главной компоненты не хватает для того, чтобы выразить всю дисперсию данных. Поэтому строят вторую, третью и так далее главные компоненты, пока их сумма не отразит всю дисперсию.

Обязательными условиями факторного анализа являются:

– признаки должны быть количественными;

– выборка должна быть однородна;

– исходные переменные должны быть распределены симметрично;

– число признаков должно быть в два раза больше числа переменных;

– анализ осуществляется по коррелирующим переменным.

1.3 Подготовка данных для анализа

Data Mining – это процесс получения из большого количества не отсортированных данных практически полезных, необходимых для каких-либо сфер деятельности человека [14]. Анализ с использованием механизмов Data Mining основывается на множестве алгоритмов получения закономерностей из исходных данных [20]. Алгоритмов достаточно много, но они не гарантируют нам качественные решения для каждой задачи, ведь методы не являются универсальными. Для каждого метода важно качество и объем исходных данных [8].

Общая схема использования методов Data Mining содержит в себе несколько шагов:

– гипотеза или предположение;

– сбор и систематизация данных;

– подбор метода, для обработки данных;

– тестирование;

– использование.

Гипотезой считают предположение о влияние каких-либо факторов на исследуемую нами задачу. Наша задача – это поиск аналогов на веб-странице, поэтому мы предполагаем, что на странице находится элемент, аналогичный выбранному.

1.4 Описание предметной области

1.4.1 Веб-скрейпинг

Web Scraping (веб-скрейпинг) – это метод используемый для извлечения с веб-сайтов большого количества ценных данных и их сохранения в отсортированном виде: в файле, таблице или базе данных (рисунок 1.7).

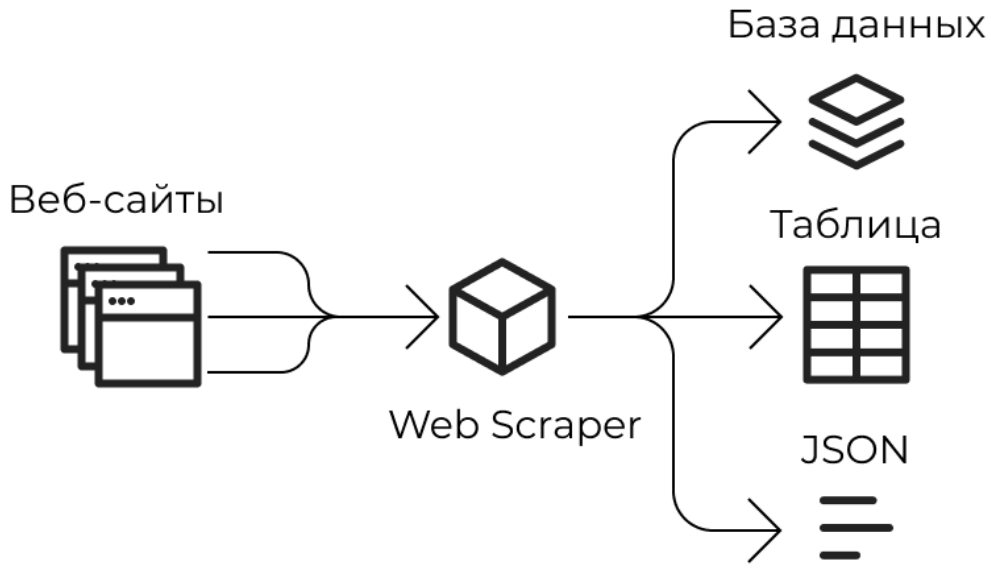

Рисунок 1.7 – Схема работы Web Scraper

Информация, отображаемая на большинстве веб-сайтов, может быть просмотрена только с помощью веб-браузера. Сайт чаще всего не имеет функции для сохранения копии размещенных на нем материалов, которые можно применять для личного использования. Единственный способ – это ручное копирование данных, что является довольно утомительной работой, выполнение которой может занять много времени и сил. Web scraping – это способ автоматизации такого процесса, выполняющий действия намного быстрее и эффективнее [2, 38–40].

Такой инструмент автоматически загрузит и извлечет информацию с одной или нескольких страниц веб-сайта, в зависимости от ваших предпочтений. Можно написать программу, которая будет сконфигурирована под единственный сайт, либо сделать ее более универсальной.

При создании программного обеспечения, способного работать с разными сайтами, нужно решить проблему поиска аналогичных элементов на веб-странице [10, 11]. Для этого нужно собрать данные, проанализировать их и протестировать на нескольких сайтах.

1.4.2 Объектная модель документа

DOM (Document Object Model – «объектная модель документа») – это дерево веб-страницы. В DOM находится вся логическая структура (представленная в виде иерархии) документа и способ с ней взаимодействовать [24]. Такое дерево позволяет программистам и пользователям намного проще перемещаться по документу и манипулировать его элементами (рисунок 1.8).

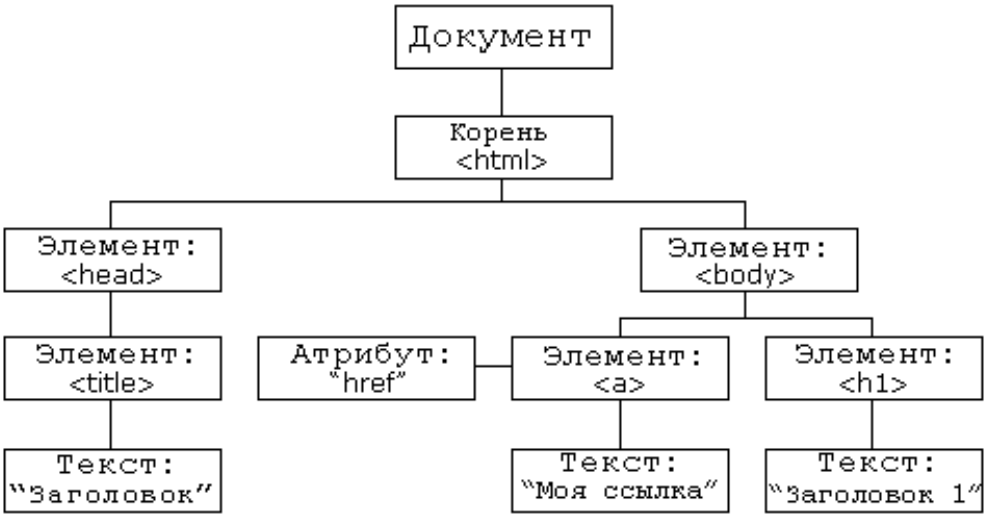

Рисунок 1.8 – Объектная модель веб-страниц

Объектная модель документа состоит из узлов, которые содержат в себе всю информацию об элементе: его параметры, родитель и дети. Такие узлы и станут нашими исходными данными, но в них есть множество ненужных для нас полей, поэтому нам нужно выбрать только самые важные.

Современные веб-страницы содержат описание разметки на языке HTML, который интерпретируется браузерами и отображается на экране компьютера или мобильного устройства [13]. Основой HTML-страниц являются теги, которые представляют собой аббревиатуры английских слов, заключенные в угловые скобки и имеют в себе значение отвечающее за вывод определенных данных на странице. Все теги имеют определенную структуру и атрибуты, которые могут изменять способ отображения содержимого и другие параметры (рисунок 1.9).

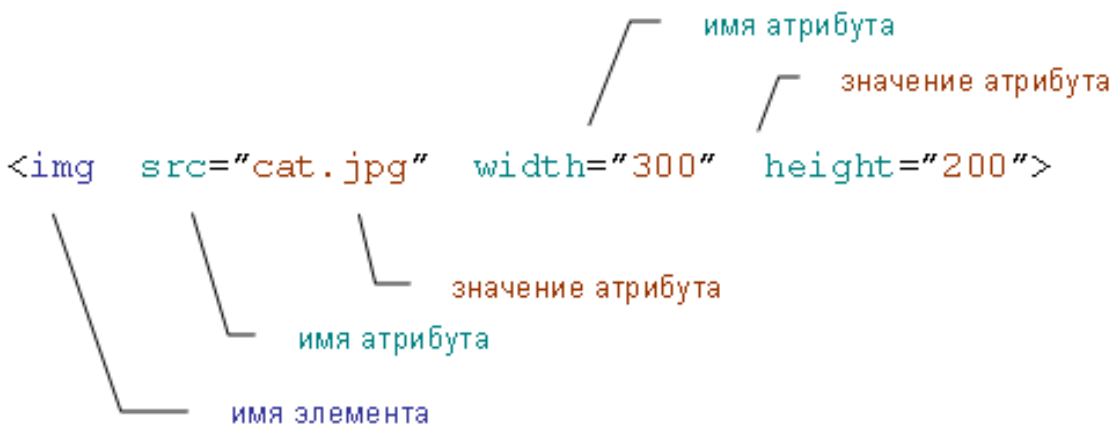

Рисунок 1.9 – Структура HTML-тега

В соответствии с DOM-деревом, каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Любой текст, который находится внутри тега, тоже является объектом. В данной ситуации, объектная модель документа – это представление HTML-документа в виде дерева тегов (рисунок 1.10).

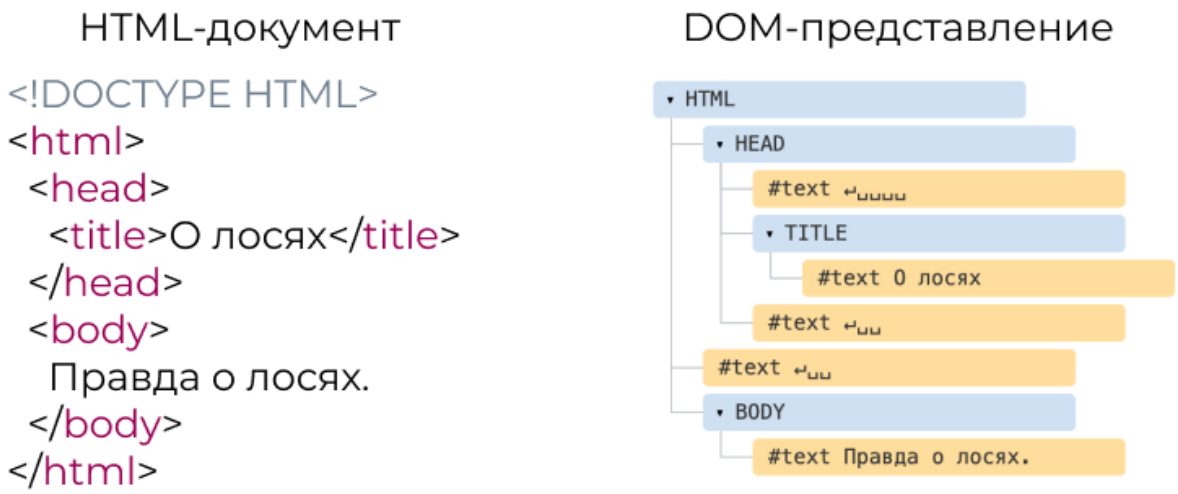

Рисунок 1.10 – DOM-представление HTML-документа

Существует двенадцать типов узлов DOM, но в основном работают с четырьмя из них:

– document – «начальная точка» дерева;

– узлы-элементы – основные строительные блоки (HTML-теги);

– текстовые узлы – содержат текст;

– комментарии – информация, которая не будет показана.

Работать с объектной моделью документа можно с помощью языка программирования JavaScript. Операции начинаются с объекта document, из которого мы можем получить доступ к любому узлу. У разных DOM-узлов могут быть разные свойства. Например, у узла, соответствующего тегу <input>, есть свойства, связанные с полем ввода, а у тега <a> связанные со ссылками, но при этом у всех элементов есть общие свойства [15]. Основными из них являются:

– children – содержит список всех детей элемента;

– parentNode –родитель элемента;

– nextSibling, previousSibling – следующий и предыдущий сосед;

– localName, tagName – полное имя тега;

– id – идентификатор элемента;

– classList – содержит список классов элемента.

Обычный пользователь, используя веб-браузер, чаще всего может визуально выделить класс элемента и его тег, другие свойства скрыты. В ходе рассмотрения нескольких элементов разных веб-страниц, для задачи поиска аналогов элемента на странице были выбраны такие: тег, идентификатор, класс, количество потомков, класс родителя, идентификатор родителя, тег родителя. Все выбранные параметры узла, кроме количества потомков, являются строками.

2 ПОСТРОЕНИЕ ПРИЗНАКОВОГО ПРОСТРАНСТВА

2.1 Постановка задачи

Необходимо построить признаковое пространство, которое поможет классифицировать каждый элемент на странице, и выбрать метод для поиска аналогов этих элементов на веб-страницах.

Для этого нужно сформировать обучающую выборку, выбрать метод для поиска аналогов и выделить оптимальное пространство признаков, выявив из всех возможных признаков благоприятные и подобрать эффективный метод, способный работать с ними. После этого, нужно проверить работу выбранного метода на тестовой выборке.

2.2 Формирование обучающей выборки

Обучающая выборка должна охватывать множество разных задач, поэтому проверить работу метода нужно на различных веб-страницах. Поэтому было выбрано несколько сайтов, представленные в таблице 1, с которых будет собираться информация.

Таблица 2 – Обучающая выборка

Ссылка	Выбираемые элементы
https://pikabu.ru/	Заголовок, текст, ссылка и автор новости
https://vk.com/team	Количество положительных оценок, изображение и текст новости
https://www.dns-shop.ru/catalog/	Изображение категории и название
https://www.ozon.ru/highlight/41676/	Изображение, цена, оценка и название товара
https://www.lamoda.ru/c/17/shoes-men/	Ссылка на товар, цена и его название
https://scotch.io/	Изображение, текст и заголовок новости

Продолжение таблицы 1

https://www.instagram.com/instagram/	Фотография, описание, количество положительных оценок
https://www.facebook.com/Facebookfor Developers/	Текст новости, изображение
https://twitter.com/search?q="В России"	Текст, имя и ссылка на автора
https://www.google.com/search?q=test	Заголовок и ссылка на сайт
https://2ch.hk/pr/	Заголовок, изображение и ссылка
https://habr.com/ru/	Заголовок новости, текст, изображение, ссылка
https://store.steampowered.com/ search/?filter=topsellers	Название игры, ссылка, цена
https://yandex.ru/search/?text=test	Заголовок, описание, ссылка
https://www.kinopoisk.ru/lists/top250/	Название фильма, постер, рейтинг
https://ru.stackoverflow.com/	Заголовок, текст вопроса, текст ответа
https://yandex.ru/images/	Изображение
https://www.reddit.com/	Заголовок новости, текст, изображение
https://hh.ru/search/	Заголовок, текст, заработная плата, компания, ссылка
https://www.youtube.com/feed/trending	Изображение, название и количество просмотров видео

Получилось 20 сайтов и 60 различных элементов на них. Данный набор имеет множество элементов, которые чаще всего встречаются на веб-страницах. Это позволит оценить универсальность выбранного метода и его точность.

2.3 Пример признакового пространства для одной из веб-страниц

Рассмотрим все возможные значения признаков одной из возможных веб-страниц. Это позволит выбрать оптимальные признаки для признакового пространства. Для этого составим таблицу, содержащую в себе каждый элемент и соответствующие ему признаки (рисунок 2.1).

	A	B	C	D	E	F	G
1	localName	id	class	childCount	parentNodeC	parentNodeL	parentNodeIc
2	html			2			
3	head			23		html	
4	style			0		head	
5	script			0		head	
6	script			0		head	
7	script			0		head	
8	script	facebook-jssdk		0		head	
9	script			0		head	
10	script			0		head	
11	script			0		head	
12	script			0		head	
13	script			0		head	
14	title			0		head	
15	meta			0		head	
16	meta			0		head	
17	meta			0		head	
18	meta			0		head	
19	link			0		head	
20	meta			0		head	
21	link			0		head	
22	link			0		head	
23	link			0		head	
24	script			0		head	
25	style			0		head	
26	style			0		head	
27	body			9		html	
28	noscript			0		body	
29	header		navbar navba	1		body	
30	div		container	2	navbar navba	header	
31	div		navbar-head	2	container	div	

Рисунок 2.1 – Таблица параметров элементов страницы

Для того, чтобы узнать насколько часто каждое значение параметра встречается в элементах одной из веб-страниц, составим гистограммы для нескольких из параметров (рисунок 2.2).

Рассмотрев гистограммы, можно сказать, что большинство параметров повторяются достаточно редко, по сравнению с пустыми значениями, поэтому данные параметры являются идеальным выбором для поиска аналогичных или элементов на странице, либо для выбора одного уникального элемента.

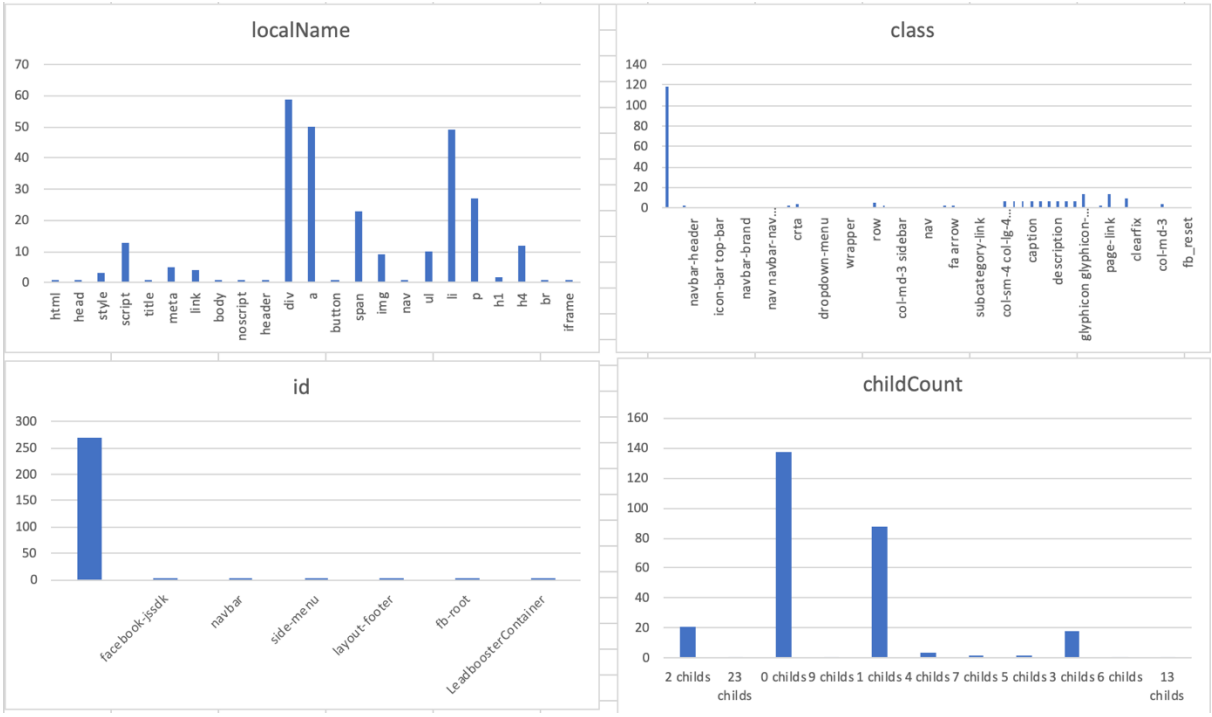

Рисунок 2.2 – Гистограммы параметров

2.4 Выбор метода для поиска аналога

Каждый из методов машинного обучения, описанных в пункте 1.1 нужно применить к рассматриваемой задаче. Оптимальным методом станет тот, который окажется наиболее быстрым и точным.

2.4.1 Деревья решений

Для построения дерева решений, рассмотрим один элемент на странице. Он имеет множество признаков, при этом каждый из них имеет множество различных значений. Это может привести к построению огромного дерева.

Возьмем, к примеру, такой параметр как класс элемента. Его значение может достигать сотню различных слов, либо наборов слов. Составив дерево только для класса элемента, получим такую же сотню узлов (рисунок 2.3).

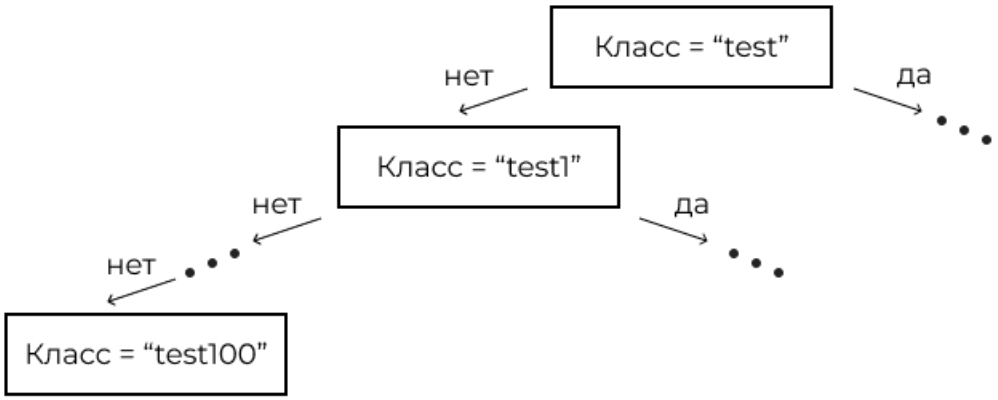

Рисунок 2.3 – Узлы дерева

При перемещении по большому дереву, выполняется огромное количество вычислений и тратится время. Применив данный метод к одной из страниц выборки, наибольшая длина пути от корня к узлу равна 1096, а поиск одного аналогичного элемента занимал более 6 секунд. Данный метод оказался совершенно не оптимален.

2.4.2 Наивный байесовский классификатор

Рассмотрим набор данных, который описывает элементы веб-страницы (таблица 2). Учитывая выбран ли элемент пользователем, каждый кортеж классифицирует условия как подходящие или непригодные для определение подобного элемента.

Таблица 2 – Набор данных для классификатора

Тэг	Класс	Количество детей	Выбран
div	thumbnail	3	Да
div	thumbnail	3	Да
div	test	2	Нет
div	test	2	Нет
a	hh	3	Нет
a	hh	3	Нет
a	hh	3	Нет
div	thumbnail	3	Нет

В данном наборе, первые два элемента выбраны пользователем, а последний является подобный им.

Используя этот набор данных, теорему Байеса можно сформулировать таким образом:

$$p(y|X) = \frac{p(X|y)p(y)}{p(x)}, \tag{2.1}$$

где y – переменная класса;

$X = (x_1, x_2, x_3, \dots, x_n)$ – вектор зависимых параметров.

Примером вектора параметров и соответствующей переменной класса может послужить третья строка:

$$X = (div, test, 2); y = \text{Нет} \tag{2.2}$$

Таким образом, $p(X|y)$ здесь обозначает вероятность того, что это не подобный элемент, учитывая что тэг «div», класс – «test» и количество детей равно трем.

	name						class			
	yes	no	P(yes)	P(no)			yes	no	P(yes)	P(no)
div	2	3	2/2	3/6		thumbnail	2	1	2/2	1/6
a	0	3	0/2	3/6		test	0	2	0/2	2/6
TOTAL	2	6				hh	0	3	0/2	3/6
						TOTAL	2	6		
	child									
	yes	no	P(yes)	P(no)			selected	selected		
3	2	4	2/2	4/6		YES	2	2/8		
2	0	2	0/2	2/6		No	6	6/8		
TOTAL	2	6				TOTAL	8			

Рисунок 2.4 – Таблицы вероятности

Для поиска аналогичных элементов, приходится выполнять множество лишних вычислений для применения теоремы Байеса. Применив данный метод к одной из страниц выборки, поиск похожих элементов занимал более 4 секунд, что является хорошим показателем по сравнению с деревом решений.

2.4.3 Выбор общих параметров

Для данного метода нужно несколько множеств признаков, выбранных пользователем. Получив их пересечение, можно искать похожие элементы на странице.

Используя набор данных из таблицы 2 можно описать такое признаки: тэг элемента, класс и количество детей. Пользователь, выбирая первые два элемента, получает два множества их параметров, а так же множество их пересечения (рисунок 2.5).

Рисунок 2.5 – Общие параметры

Рассмотрев метод, можно сказать, что он делает то же самое, что и метод выбора общих параметров, но выполняя множество ненужных вычислений, поэтому наиболее оптимальным для рассматриваемой задачи станет метод выбора общих параметров.

Поиск аналогичных элементов на странице из выборки занял менее 1 секунды, поэтому данный метод является наиболее оптимальным из рассмотренных.

Главным критерием в точности его работы станет признаковое пространство, которое создается для каждого элемента, поэтому выбор правильного набора признака является важной задачей.

2.5 Отбор оптимальных признаков

Используя метод выбора общих параметров и обучающую выборку попробуем выделить набор оптимальных признаков. Для этого возьмем все свойства элемента страницы и попробуем выполнить поиск элементов на странице используя созданное методом признаковое пространство.

Было получено, что данное пространство работает в 96 процентах случаев (рисунок 2.6).

	A	B	C	D	E	F
1	**Ссылка**	**Выбираемые элементы**	**Все**	**Выбранные**	**ИТОГ**	
2	https://pikabu.ru/	заголовок	1	1	**Все**	**Выбранные**
3	https://pikabu.ru/	текст	1	1	0,95	0,95
4	https://pikabu.ru/	ссылка	1	1		
5	https://pikabu.ru/	автор	1	1		
6	https://vk.com/team	положительные оценки	1	1		
7	https://vk.com/team	изображение	1	1		
8	https://vk.com/team	текст	1	1		
9	https://www.dns-shop.ru	изображение	1	1		
10	https://www.dns-shop.ru	название	1	1		
11	https://www.ozon.ru/hig	изображение	1	1		
12	https://www.ozon.ru/hig	цена	1	1		
13	https://www.ozon.ru/hig	оценка	1	1		
14	https://www.ozon.ru/hig	название	1	1		
15	https://www.lamoda.ru/	ссылка	1	1		
16	https://www.lamoda.ru/	цена	1	1		
17	https://www.lamoda.ru/	название	1	1		
18	https://scotch.io/	изображение	1	1		
19	https://scotch.io/	текст	1	1		
20	https://scotch.io/	заголовок	1	1		
21	https://www.youtube.cor	изображение	1	1		
22	https://www.youtube.cor	название	1	1		
23	https://www.youtube.cor	количество просмотров	1	1		
24	https://www.instagram.c	фотография	0	0		
25	https://www.instagram.c	описание	0	0		
26	https://www.instagram.c	ичество положительных оце	0	0		
27	https://www.facebook.cc	текст	1	1		
28	https://www.facebook.cc	изображение	1	1		
29	https://twitter.com/sear	текст	1	1		
30	https://twitter.com/sear	имя	1	1		
31	https://twitter.com/sear	ссылка	1	1		
32	https://www.google.com	заголовок	1	1		
33	https://www.google.com	ссылка	1	1		
34	https://2ch.hk/pr/	заголовок	1	1		
35	https://2ch.hk/pr/	изображение	1	1		
36	https://2ch.hk/pr/	ссылка	1	1		

Рисунок 2.6 – Результаты тестирования

Для построения оптимального признакового пространства нужно выбрать только самые важные признаки элемента.

В пункте 1.3.2 были описаны основные свойства элементов. Нужно изменить данные свойства так, чтобы была возможность их сравнивания. Для этого нужно избавиться от полей, хранящие в себе объекты, так как это может усложнить наши вычисления.

Мы избавимся от двух полей nextSibling и previousSibling, так как эти свойства, скорей всего не совпадут у аналогичных элементов. Например, элемент может находится в контейнере, либо перед первым из них будет находится картинка, что делает это поле бесполезным.

Свойство children возвращает список всех потомков, поэтому это усложняет вычисления. Если даже сравнивать каждого «ребенка» элемента с потомками другого элемента, они не будет полностью равны, так как являются аналогичными элементами, но не копиями. Поэтому заменим данное поле на childrenCount – число детей элемента. Такое поле легко сравнивается и должно быть одинаковым для похожих элементов.

Такое свойство как parentNode нужно поделить на несколько. Нужно знать всего лишь класс предка, его тэг и идентификатор.

Поле classList, также как и children, возвращает список классов, присвоенных элементу. Заменим список простой строкой, где классы указаны через пробел. Это возможно, так как в CSS названия классов не имеют такого разделительного знака как пробел.

Выбирая из свойств элемента tagName и localName, выберем второй, потому что первый метод, возвращает имя тега в верхнем регистре, что может повлиять на результаты распознавания.

Свойство innerText, содержит в себе весь текст, который находится в элементе, поэтому данный параметр позволит выбирать уникальные элементы.

Остальные свойства остаются без изменений. В итоге получим такие признаки элемента на веб-странице:

– id – идентификатор элемента;

– class – класс элемента;

– localName – имя тэга элемента;

– childCount – количество потомков;

– parentNodeClass – класс родителя;

– parentNodeLocalName – имя тэга родителя;

– parentNodeId – идентификатор родителя;

– innerText – текст внутри элемента.

Проведем анализ зависимости признаков друг от друга. Для этого составим для каждой пары признаков соответствующую таблицу сопряженности (рисунок 2.7).

	facebook-jssdk	navbar	side-menu	layout-footer	fb-root	LeadboosterContainer		ВСЕГО	q
html	0	0	0	0	0	0	1	1	0,003623188
head	0	0	0	0	0	0	1	1	0,003623188
style	0	0	0	0	0	0	3	3	0,010869565
script	1	0	0	0	0	0	12	13	0,047101449
title	0	0	0	0	0	0	1	1	0,003623188
meta	0	0	0	0	0	0	5	5	0,018115942
link	0	0	0	0	0	0	4	4	0,014492754
body	0	0	0	0	0	0	1	1	0,003623188
noscript	0	0	0	0	0	0	1	1	0,003623188
header	0	0	0	0	0	0	1	1	0,003623188
div	0	0	0	1	1	0	57	59	0,213768116
a	0	0	0	0	0	0	50	50	0,18115942
button	0	0	0	0	0	0	1	1	0,003623188
span	0	0	0	0	0	0	23	23	0,083333333
img	0	0	0	0	0	0	9	9	0,032608696
nav	0	1	0	0	0	0	0	1	0,003623188
ul	0	0	1	0	0	0	9	10	0,036231884
li	0	0	0	0	0	0	49	49	0,177536232
p	0	0	0	0	0	0	27	27	0,097826087
h1	0	0	0	0	0	0	2	2	0,007246377
h4	0	0	0	0	0	0	12	12	0,043478261
br	0	0	0	0	0	0	1	1	0,003623188
iframe	0	0	0	0	0	1	0	1	0,003623188
ВСЕГО	1	1	1	1	1	1	270	276	
p	0,003623188	0,00362319	0,003623188	0,003623188	0,003623188	0,003623188	0,97826087		

Рисунок 2.7 – Таблица сопряженности двух признаков

Допусти, что эти признаки являются независимыми и построим статистику χ^2.

Таблица 3 – Значения χ^2и $\chi^2_{\text{крит}}$ для двух соответствующих признаков

	localName	id	class	childCount
localName		0,0023 / 97,1634	0,2535 / 1203,4189	0,7208 / 174,1603
id	0,0023 / 97,1634		0,0006 / 300,5325	0,0032 / 37,4849
class	0,2535 / 1203,4189	0,0006 / 300,5325		0,0659 / 522,36512
childCount	0,7208 / 174,1603	0,0032 / 37,4849	0,0659 / 522,3651	

Значение χ^2для каждого признака значительно отличается от $\chi^2_{\text{крит}}$. Из этого следует, что все признаки являются независимыми друг от друга.

Применив данное признаковое пространство к методу выбора общих параметров, получим результат в 95%. Это обозначает, что данное пространство является более оптимальным для рассматриваемой задачи.

Пять процентов элементов, не прошедших проверку, находятся на сайте компании Instagram. Данная компания борется с людьми, которые пытаются собирать фотографии с их сайта при помощи изменения всех идентификаторов элементов в DOM-дереве при каждой перезагрузке страницы. Такие действия вводят в заблуждение программы по сбору информации и не дают автоматизировать этот процесс. Целесообразно рассмотреть признаки, которые формируют местоположение элемента на странице, его внешний вид, а так же признаки относящиеся к структуре страницы.

2.6 Выводы по разделу

После сравнения нескольких методов машинного обучения, было принято решение о применении метода выбора общих параметров. Данный метод оказался прост в реализации для данной задачи.

В ходе подбора признаков и метода поиска аналогов на веб-странице, было выбрано несколько подходящих признаков. Они являются простыми в сравнении, и не требуют больших вычислительных затрат для получения. Данный набор признаков был опробован на обучающей выборке и дал хороший результат.

Стало понятно, что существую сайты, которые борются с программами для автоматического сбора информации, но в большинстве случаев сбор происходит без проблем.

3 РАЗРАБОТКА ПРОГРАММЫ ДЛЯ ТЕСТИРОВАНИЯ МЕТОДА ПОИСКА АНАЛОГОВ

3.1 Разработка архитектуры приложения

Web Scraping можно разделить на две категории: сбор данных используя программное обеспечение, либо с помощью написания кода. Первый способ универсальный для многих сайтов и удобен для большинства людей. Второй, используется для получения информации с определенного сайта используя API (программный интерфейс приложения), либо элементы веб-страниц [25].

Самой большой и главной проблемой в программном обеспечении для web scraping является сложность в его настройке и использовании, поэтому разработчики пытаются решить эту проблему, разрабатывая удобные для пользователя программы [22].

Сбор информации может происходить на компьютере пользователя, либо на удаленном сервере. Второй метод является хорошей альтернативой, когда программа выполняет большое количество вычислений, требующих больших компьютерных мощностей. При выполнении запросов от множества пользователей к сайту, сервер может быть заблокирован из-за подозрительной активности, ведь обычный пользователь не выполняет такое количество переходов по сайту. Сбор данных при помощи одного компьютера уменьшает шанс того, что программа будет заблокирована сайтом, поэтому сбор будет производиться в веб-браузере пользователя.

Так как мы работаем в браузере, платформой для разработки программного обеспечения были выбраны расширения – это программы, расширяющие возможности веб-браузера.

Разработка будет вестись под Google Chrome, так как это самый популярный браузер, имеющий все актуальные возможности современных браузеров и позволяющий расширить свой функционал.

Любое Chrome расширение чаще всего имеет довольно простую структуру [27]:

• manifest.json – главный файл, в котором хранится информация о подключаемых файлах, настройках безопасности и т. д.;

• background script – невидимая страница которая содержит основную логику расширения;

• content script – скрипты которые выполняются в контексте веб-страницы;

• popup – страница отображаемая при нажатии на иконку расширения;

• devtools – инструменты разработчика.

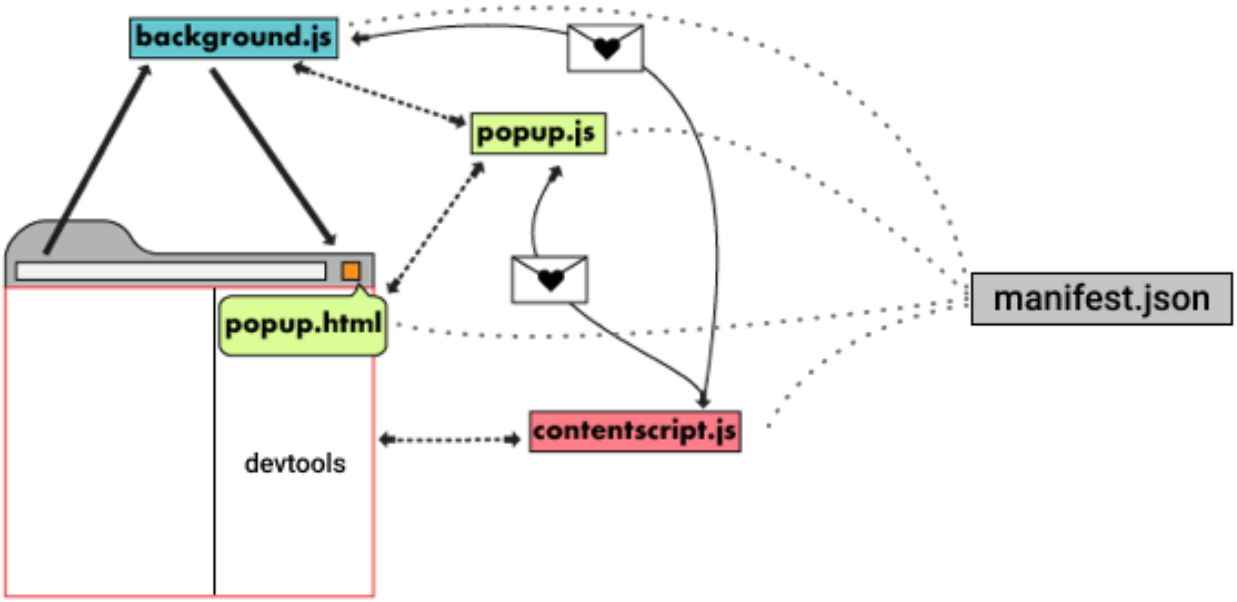

Рисунок 3.1 – Структура расширения Google Chrome

Полноценный доступ к DOM дереву есть только у content скрипта, так как именно он напрямую взаимодействует со страницей.

Общение между скриптами в расширении происходит посредством JavaScript событий. Один файл отправляет события, а другой принимает их и обрабатывает. Таким образом можно выполнять функции на веб-странице, выполняя действия во всплывающем окне расширения [26].

Интерфейс был разработан с помощью JavaScript-библиотеки React [36]. Данная библиотека предоставляет высокую скорость работы, простоту, масштабируемость и позволяет удобно создавать интерфейс приложения,

используя компонентный подход. Она поможет быстро описать компоненты расширения и указать на логику их взаимодействия между собой.

Компоненты позволяют разделить пользовательский интерфейс на независимые, повторно используемые части и работать с ними отдельно. Они принимают произвольные данные и возвращают React-элементы, которые описывают, что должно появиться на экране. Для расширения было выделено несколько компонентов, которые будут повторятся во всем приложении и позволят удобно работать с запросами пользователя (рисунок 3.2):

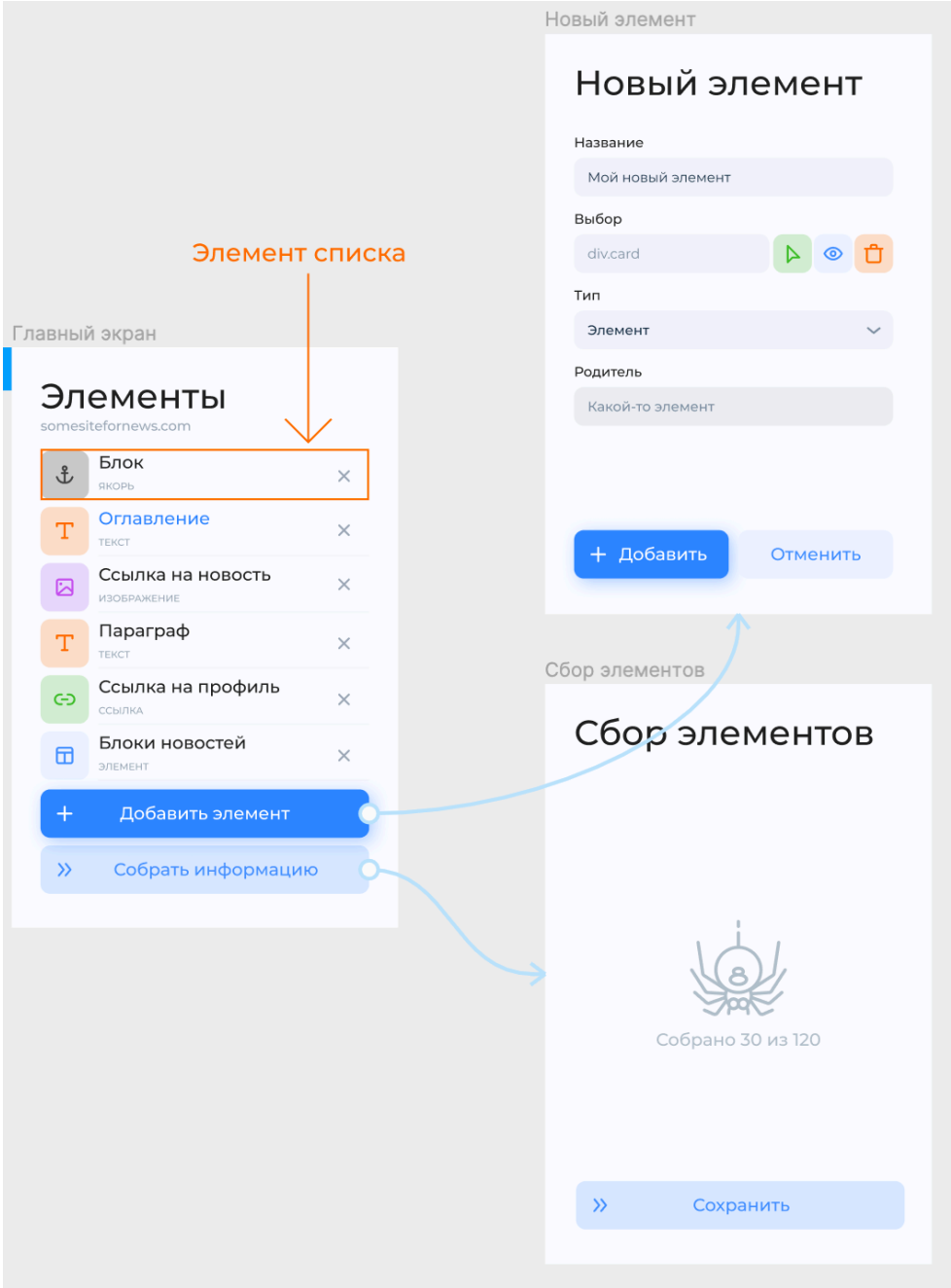

Рисунок 3.2 – Компоненты расширения

1. Главный экран – содержит список созданных элементов, кнопки добавления новых элементов и кнопки сбора информации.

2. Экран добавления/изменения элемента – имеет поля для ввода названия, выбора типа и места на странице.

3. Сбор информации – содержит прогресс сбора информации и кнопку для ее сохранения.

4. Элемент списка – включает название элемента, его тип, иконка и действия.

Расширение будет находиться в инструментах разработчика Google Chrome, что позволит удобно выбирать элементы на странице и скроет приложение во время повседневного использования браузера. Исходный код расширения расположен в приложении 1.

Во время разработки приложения было выделено несколько типов элементов:

- элемент – является контейнером для текста, изображения и ссылки;
- якорь – элемент, который должен встретится на каждой странице;
- текст – содержит в себе текст элемента;
- изображение – содержит в себе ссылку на изображение;
- ссылка – содержит ссылку элемента.

3.2 Проверка корректности работы

Рассмотрим несколько сайтов и попробуем выбрать на них похожие элементы, для сбора информации, находящейся в них.

Для этого заходим на страницу для проверки и запускаем расширение. Оно находится в инструментах разработчика браузера Google Chrome в соответствующей вкладке. При запуске программы, на главном экране расширения должно отобразить домен веб-сайта в заголовке, а так же кнопки для сбора информации и добавления новых элементов (рисунок 3.3).

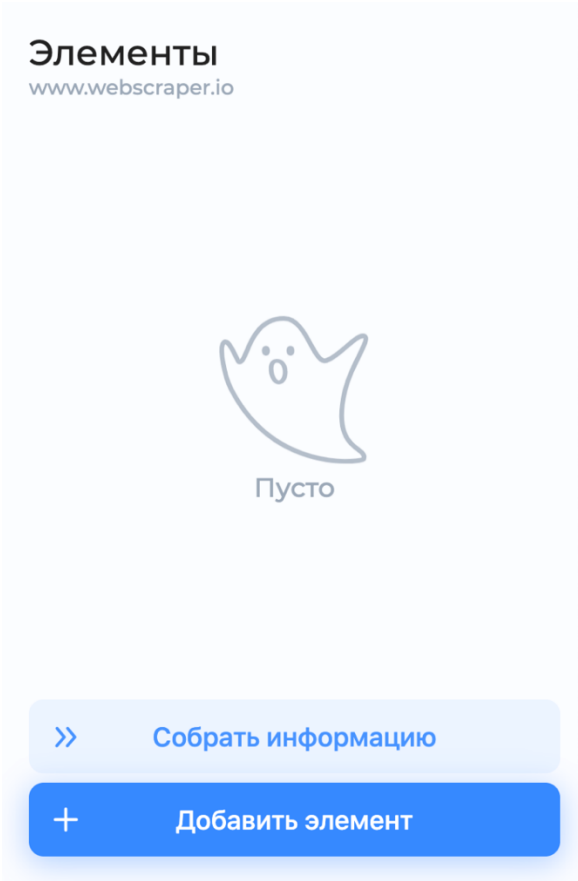

Рисунок 3.3 – Главное окно расширения

Домен отобразился. Следующим действием будет добавление элемента со страницы. Для этого нажимаем на кнопку «Добавить элемент», после чего вводим название элемента (используется как заголовок таблицы с данными), который мы собираемся выбрать и нажимаем на иконку курсора (рисунок 3.4).

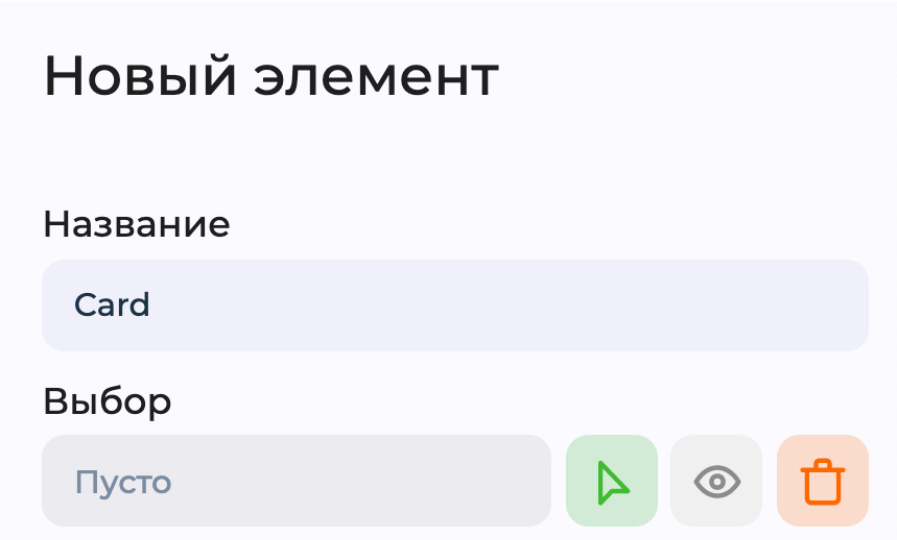

Рисунок 3.4 – Поле названия и выбора элемента

После чего на сайте выбираем пару элементов. Эти действия должны привести к тому, что на странице выделятся красным выбранные элементы (рисунок 3.5).

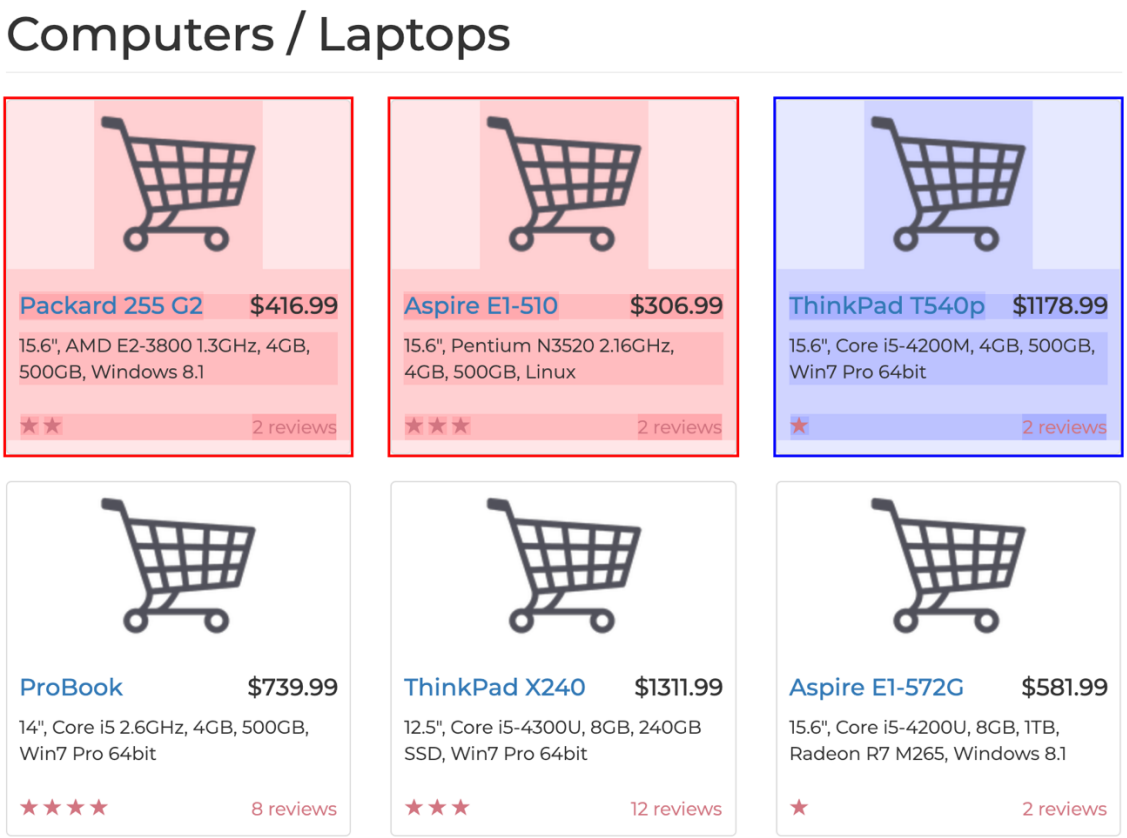

Рисунок 3.5 – Выбор элементов

Нужно вернуться в расширение и нажать на иконку галочки. В итоге поле «Выбор» будет выводить тэг и класс аналогичных элементов на странице (рисунок 3.6).

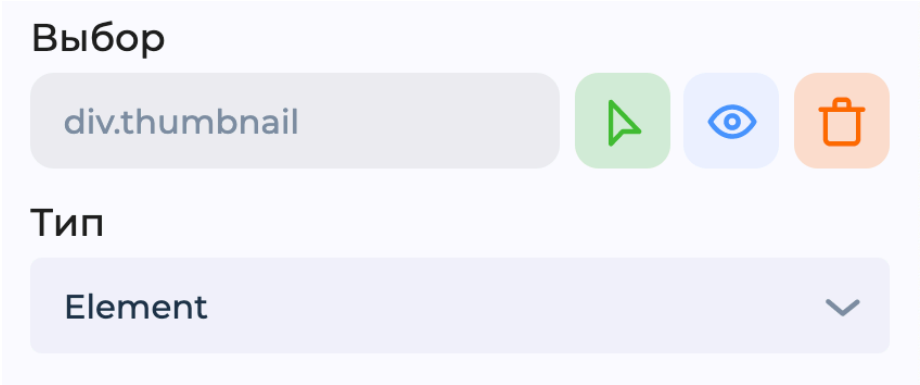

Рисунок 3.6 – Поле «Выбор»

Можно еще раз посмотреть на выбранные аналоги. Для этого нужно нажать на иконку глаза, после чего на текущей странице отобразятся элементы, которые являются аналогичными выбранным. По желанию, можно нажать на иконку корзины, тогда параметр очистится.

Теперь можно выбрать тип элемента. Элементы могут быть пяти типов: элемент, якорь, ссылка, изображение, текст. Данный параметр поможет правильно получить данные из выбранных элементов, а также удобно их отсортировать. Элемент и якорь могут быть выбранными только на первом уровне, так как они выделяют сам элемент на странице как контейнер. Остальные типы нужны для схематического выбора частей каждого из них, при этом родительский элемент подсвечивается оранжевым (рисунок 3.7).

Рисунок 3.7 – Выбор частей элемента

Составив схему для элементов страницы, пользователь возвращается на главный экран расширения и нажимает на кнопку «Собрать информацию», что запускает процесс сбора информации и отображает количество найденных элементов на веб-странице (рисунок 3.8).

Рисунок 3.8 – Сбор элементов

В результате сбора информации, пользователю предлагается сохранить собранную информацию в CSV файле. Данных файл будет содержать таблицу, где первая строка – это название каждой части, описанной в расширении перед сбором, а другие строки – значение соответствующее каждому элементу (рисунок 3.9).

	A	B	C	D	E
1	Заголовок	Ссылка	Цена	Описание	
2	Packard 255 G2	https://www.webscraper.io/test-sites/e	$416.99	15.6", AMD E2-3800 1.3GH	
3	Aspire E1-510	https://www.webscraper.io/test-sites/e	$306.99	15.6", Pentium N3520 2.16	
4	ThinkPad T540p	https://www.webscraper.io/test-sites/e	$1178.99	15.6", Core i5-4200M, 4GB	
5	ProBook	https://www.webscraper.io/test-sites/e	$739.99	14", Core i5 2.6GHz, 4GB, !	
6	ThinkPad X240	https://www.webscraper.io/test-sites/e	$1311.99	12.5", Core i5-4300U, 8GB,	
7	Aspire E1-572G	https://www.webscraper.io/test-sites/e	$581.99	15.6", Core i5-4200U, 8GB,	
8	Aspire D3-320	https://www.webscraper.io/test-sites/e	$306.99	15.6", Pentium N3520 2.16	
9	Aspire D3-570	https://www.webscraper.io/test-sites/e	$581.99	15.6", Core i5-4200U, 8GB,	
10	Packard 305 G3	https://www.webscraper.io/test-sites/e	$416.99	15.6", AMD E2-3800 1.3GH	
11	ProBook2	https://www.webscraper.io/test-sites/e	$739.99	14", Core i5 2.6GHz, 4GB, !	
12	ThinkPad T530	https://www.webscraper.io/test-sites/e	$1178.99	15.6", Core i5-4200M, 4GB	
13	ThinkPad 300	https://www.webscraper.io/test-sites/e	$1311.99	12.5", Core i5-4300U, 8GB,	

Рисунок 3.9 – Результат работы программы

Рассматривая одну из страниц, данная программа нашла 120 аналогичных элементов и собрала информацию с них за 6 секунд. Программа переходила по всем ссылкам, ведущим на этот же сайт. Ссылки добавлялись рекурсивно при условии, что на странице находится элемент «якорь». Это позволяет не собирать ненужные элементы.

3.4 Выводы по разделу

В ходе разработки архитектуры приложения и тестирования его функционала, были получены промежуточные результаты. Выбранный метод и построенное признаковое пространство оказались удачными. Программа работает во всех проверенных случаях и правильно определяет аналогичные элементы.

С помощью программа было найдено 120 аналогичных элементов на всем сайте. Они были собраны всего за 6 секунд, что является наиболее оптимальным методом по сравнению с ручным сбором информации.

ЗАКЛЮЧЕНИЕ

Рассмотрена задача построения признакового пространства для элементов веб-страниц и разработана программное обеспечение в форме расширения для браузера.

Выполнен обзор существующих методов поиска аналогов и анализа признакового пространства. Было рассмотрено три метода: дерево решений, наивный байесовский классификатор и метод выбора общих параметров. Метод выбора общих параметров показал наилучшие результаты.

Было выбрано несколько признаков для элементов веб-страниц из всех признаков описанных в DOM-дереве элемента. Произведена проверка на независимость признаков. Все выбранные признаки оказались независимыми. Данное признаковое пространство позволило выбрать аналоги на страницах из обучающей выборки в 95% случаев.

Была разработана программа на языке JavaScript как расширение для браузера Google Chrome и протестирована на одном из сайтов из выборки. Программа нашла все 120 аналогичных элементов на сайте и собрала их за 6 секунд.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Бабенко, Д. Алгоритмы интеллектуального интернета. Передовые методики сбора, анализа и обработки данных / Д. Бабенко; пер. с англ. М.А. Низовец. – СПб.: Символ-Плюс, 2011. – 480 с.

2 Басалаева, А.Ю. Web-scraping и классификация текстов методом наивного Байеса / А.Ю. Басалаева, Г.А. Гареева, Д.Р. Григорьева // Инновационная наука. 2018. №5. URL: https://cyberleninka.ru/article/n/web-scraping-i-klassifikatsiya-tekstov-metodom-naivnogo-bayesa (дата обращения: 06.05.2020).

3 Бурлаева, Е.И. Обзор методов классификации текстовых документов на основе подхода машинного обучения / Е.И. Бурлаева // Программная инженерия. – 2017. – № 7. – С. 328–336.

4 Вьюнин, В.В. Математические основы теории машинного обучения и прогнозирования / В.В. Вьюнин. – М.: МЦНМО, 2013. – 390 с.

5 Денисова, Д.С. Автоматическая обработка языка. Классификация текста. Наивный байесовский классификатор / Д.С. Денисова // Синергия наук. – 2018. – № 19. – С. 1410–1414.

6 Корреляция. Коэффициент корреляции Пирсона. – URL: http://www.machinelearning.ru/wiki/index.php?title=Коэффициент_корреляции_Пирсона (дата обращения 03.05.2020).

7 Корреляция случайных величин. Университет ИТМО. – URL: https://neerc.ifmo.ru/wiki/index.php?title=Корреляция_случайных_величин (дата обращения 03.05.2020).

8 Менщиков, А.А. Изучение поведения средств автоматизированного сбора информации с веб-ресурсов / А.А. Менщиков, А.В. Комарова, Ю.Г. Гатчин // Вопросы кибербезопасности. 2017. №3 (21). URL: https://cyberleninka.ru/article/n/izuchenie-povedeniya-sredstv-avtomatizirovannogo-sbora-informatsii-s-veb-resursov (дата обращения: 16.05.2020).

9 Мерков, А.Б. Распознавание образов. Построение и обучение вероятностных моделей / А.Б. Мерков. – СПб.: Ленанд, 2014. – 238 с.

10 Митчелл, Р. Скрапинг веб-сайтов с помощью Python / Р. Митчелл; пер. с англ. А.В. Груздев. – М.: ДМК Пресс, 2016. – 280 с.

11 Москаленко, А.А. Разработка приложения веб-скрапинга с возможностями обхода блокировок / А.А. Москаленко, О.Р. Лапонина, В.А. Сухомлин // Современные информационные технологии и ИТ-образование. 2019. № 2. URL: https://cyberleninka.ru/article/n/razrabotka-prilozheniya-veb-skrapinga-s-vozmozhnostyami-obhoda-blokirovok (дата обращения: 06.05.2020).

12 Наивные байесовские классификаторы | портал информатики для гиков. – URL: http://espressocode.top/naive-bayes-classifiers/ (дата обращения 20.04.2020).

13 Понятие тега HTML. – URL https://samsebewebmaster.ru/uroki-html-i-css/ponyatie-tega-html.html (дата обращения 28.04.2020).

14 Руководство аналитика Deductor Studio 5.3. – URL: https://basegroup.ru/deductor/manual/guide-analyst-530 (дата обращения 03.04.2020).

15 Свойства узлов: тип, тег и содержимое. – URL: http://learn.javascript.ru/basic-dom-node-properties (дата обращения 01.05.2020).

16 Симпсон, К. {Вы не знаете JS} Типы и грамматические конструкции. / К. Симпсон // СПб.: Питер, 2019. – 240 с.

17 Современный учебник JavaScript. – URL: https://learn.javascript.ru/ (дата обращения 10.04.2020).

18 Субботин, С.В. Применение байесовского классификатора для распознавания классов целей / С.В. Субботин, Д.Ю. Большаков // Журнал радиоэлектроники. – 2006. – № 4. – С. 216–236.

19 Третьяков, А.О. Метод определения русскоязычных фейковых новостей с использованием элементов искусственного интеллекта / А.О. Третьяков, О.Г. Филатова, Д.В. Жук, Н.Н. Горлушкина // International Journal of Open Information Technologies. 2018. №12. URL: https://cyberleninka.ru/article/n/metod-opredeleniya-russkoyazychnyh-feykovyh-

novostey-s-ispolzovaniem-elementov-iskusstvennogo-intellekta (дата обращения: 14.05.2020).

20 Флах, П. Машинное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах; пер. с англ. А.А. Слинкин. – М.: ДМК, 2016. – 400 с.

21 Флэнаган, Д. JavaScript. Подробное руководство, 6-е издание / Д. Флэнаган; пер. с англ. – СПБ: Символ-Плюс, 2012. – 1080 с.

22 Хашковский, В.В. Современные подходы в организации систем обработки больших объемов данных / В.В. Хашковский, А.Н. Шкурко // Известия ЮФУ. Технические науки. 2014. №8 (157). URL: https://cyberleninka.ru/article/n/sovremennye-podhody-v-organizatsii-sistem-obrabotki-bolshih-obemov-dannyh (дата обращения: 14.05.2020).

23 Шанов, С.В. Применение байесовского классификатора для определения тематики текста / С.В. Шанов // Моделирование, оптимизация и информационные технологии. – 2018. – № 1. – С. 131–139.

24 DOM-дерево. – URL: https://learn.javascript.ru/dom-nodes (дата обращения 01.05.2020).

25 Everything you Need to Know About Web Scraping. – URL: https://towardsdatascience.com/everything-you-need-to-know-about-web-scraping-6541b241f27e (дата обращения 10.04.2020).

26 Extend the Browser. – URL: https://developer.chrome.com/extensions (дата обращения: 14.04.2020).

27 Google Chrome Extensions – Часть 1. Архитектруа. – URL: https://medium.com/@aleksanderzinewicz/google-chrome-extensions-часть-1-архитектура-fcb0dc0949d4 (дата обращения: 14.04.2020).

28 Henrik, B. Machine Learning / B. Henrik, R. Joseph, M. Fetherolf. – New-York: Manning Publication Co, 2016. – 264 p.

29 Karaseva, T.S. Comparison of data mining techniques performance in spam detection / T.S. Karaseva, E.S. Semenkin // Современная наука, техника и инновации. – 2015. – № 14. – С. 31–33

30 Kuhn, M. Applied Predictive Modeling / M. Kuhn, K. Johnson. – New-York: Cambridge University, 2014. – 600 p.

31 Leskovec, J. Mining of Massive Datasets / J. Leskovec, R. Anand, D. Ullman. – California: Cambridge University, 2014. – 513 p.

32 Lomakina, L.S. Naïve bayes modification for text streams classification / L.S. Lomakina, D.V. Lomakin // Sciences of europe. – 2016. – № 6. – С. 38–41.

33 Marsland, S. Machine Learning: An Algorithmic Perspective / S. Marsland. – Florida: CRC, 2015. – 390 p.

34 Naive Bayes Classifier. – URL: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (дата обращения: 13.05.2020).

35 Patel, K.B. Process of web usage mining to find interesting patterns from web usage data / Patel K.B., Patel A.R. – Режим доступа: http://www.ijctonline.com – 2012. – Vol. 3, № 1.

36 React – A JavaScript library for building user interfaces. – URL: https://reactjs.org/ (дата обращения: 13.05.2020).

37 Theodoridis, S. Machine Learning: A Bayesian and Optimization Perspective / S. Theodoridis. – Orlando: Academic, 2015. – 1062 p.

38 The Ultimate Guide to Web Scraping with Node.js – URL: https://www.freecodecamp.org/news/the-ultimate-guide-to-web-scraping-with-node-js-daa2027dcd3/ (дата обращения: 01.05.2020).

39 Web scraping при помощи Node.js – URL: https://habr.com/ru/post/301426/ (дата обращения: 01.05.2020).

40 What is Web Scraping and How Does Web Crawling Work? – URL: http://www.scrapinghub.com/what-is-web-scraping (дата обращения: 01.05.2020).

ПРИЛОЖЕНИЕ 1
Исходный код

Файл manifest.json

```
{
  "manifest_version": 2,

  "name": "WebScraper extension",
  "description": "This extension helps to web scrap",
  "version": "0.0.5",

  "browser_action": {
    "default_popup": "index.html",
    "default_title": "Open the popup"
  },
  "devtools_page": "index.html",
  "content_scripts": [
    {
      "matches": ["<all_urls>"],
      "js":["content.js"],
      "css":["hover.css"]
    }
  ],
  "icons": {
    "16": "icon16.png",
    "32": "icon32.png",
    "48": "icon48.png",
    "64": "icon64.png",
    "128": "icon128.png",
    "192": "icon192.png",
    "256": "icon256.png"
  },
  "content_security_policy":        "script-src        'self'        'sha256-
XSZoPF6vJhDUU97e1L5IFbyL2hR1oQP9Zsli/Mv6acM='; object-src 'self'",
  "permissions": [
    "contextMenus", "tabs", "<all_urls>"
  ]
}
```

Файл index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link
href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;500&displa
y=swap" rel="stylesheet">
    <link href="index.css" rel="stylesheet">
    <title>WebScrapper</title>
    <script src="devtools.js"></script>
  </head>
```

```
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Файл index.css

```
body, html{
    padding: 0;
    margin: 0;
    height: 100%;
}
```

Файл hover.css

```
.hint-mode_show-block, .hint-mode_show-block *{
    background-color: rgba(0, 0, 255, 0.1) !important;
}

.hint-mode_show-block{
    outline: rgb(0, 0, 255) solid 2px !important;
}

.selected-mode_show-block, .selected-mode_show-block *{
    background-color: rgba(255, 0, 0, 0.1) !important;
}

.selected-mode_show-block{
    outline: rgb(255, 0, 0) solid 2px !important;
}

.check-mode_show-block{
    outline: rgb(255, 255, 0) solid 2px !important;
}

.check-mode_show-block, .check-mode_show-block *{
    background-color: rgba(255, 255, 0, 0.1) !important;
}

.check2-mode_show-block{
    outline: rgb(255, 150, 0) solid 2px !important;
}

.check2-mode_show-block, .check2-mode_show-block *{
    background-color: rgba(255, 150, 0, 0.1) !important;
}
```

Файл devtools.js

```
/*global chrome*/
chrome.devtools.panels.create("Scraper", null, "index.html", null);
```

Файл App.js

```
import React from 'react';
import './App.css';
import MainPage from './MainPage';
import ListBlock from './ListBlock';
```

```
import AddPage from './AddPage';
/*global chrome*/

class App extends React.Component {
  constructor(props){
    super(props);
    this.state = {
      showAdd: false,
      editElement: undefined,
      elements: [],
      currentSite: undefined,
      parent: undefined
    }

    this.handleToggleAddClick = this.handleToggleAddClick.bind(this);
    this.handleElementAdd = this.handleElementAdd.bind(this);
    this.handleBlockClick = this.handleBlockClick.bind(this);
    this.handleBlockDelete = this.handleBlockDelete.bind(this);
    this.handleBlockEdit = this.handleBlockEdit.bind(this);
    this.handleBackClick = this.handleBackClick.bind(this);
    this.handleScrapClick = this.handleScrapClick.bind(this);
  }

  componentDidMount() {
    chrome.tabs.query({active: true, lastFocusedWindow: true}, tabs => {
        let tmp = document.createElement ('a');
        tmp.href = tabs[0].url;
        const oldElements = localStorage.getItem(tmp.hostname)
        this.setState({
          elements: oldElements ? JSON.parse(oldElements) : [],
          currentSite: tmp.hostname
        })
    });
  }

  handleToggleAddClick(){
    this.setState(prevState => ({
      editElement: undefined,
      showAdd: !prevState.showAdd,
    }))
  }

  handleScrapClick(){
    const {elements} = this.state
    chrome.tabs.query({currentWindow: true, active: true}, function (tabs){
                  var activeTab = tabs[0];
                  chrome.tabs.sendMessage(activeTab.id,  {"message":  "scrap",
"body": elements});
            });
  }

  handleBackClick(parentId){
    const el = this.state.elements.filter((el) => el.id === parentId)
    this.setState({
      parent: el.length > 0 ? el[0] : undefined
    })
```

```
}

getDeleteElements(id){
  let delElements = []
  let ids = [id]
  this.state.elements.forEach((el, arrId)=>{
    for(let checkId of ids) {
      if(checkId === el.id){
        delElements.push(arrId)
        break;
      } else if (el.parent && el.parent.id === checkId){
        ids.push(el.id)
        delElements.push(arrId)
        break;
      }
    }
  })
  return delElements
}

handleBlockDelete(id){
  let elements = [...this.state.elements]
  const deleteElements = this.getDeleteElements(id)
  for(let el of deleteElements.reverse()){
    elements.splice(el, 1)
  }
  this.setState({
    elements
  })
  localStorage.setItem(this.state.currentSite, JSON.stringify(elements))
}

handleBlockClick(element){
  this.setState({
    parent: element,
  })
}

handleBlockEdit(element){
  this.setState((prevState) => ({
    editElement: element,
    showAdd: !prevState.showAdd,
  }))
}

getElementId(){
  const idArray = this.state.elements.map((el) => el.id)
  if(idArray.length === 0) return 0
  else return Math.max(...idArray) + 1
}

handleElementAdd(element) {
  const {elements} = this.state
  let newElements = [...elements]
  if(element.id !== undefined){
    newElements.forEach((el, arrId)=>{
```

```
        if(el.id === element.id){
          newElements[arrId] = element
          return
        }
      })
    } else {
      element.id = this.getElementId()
      newElements.push(element)
    }
    this.setState({
      elements: newElements,
      showAdd: false
    })
    localStorage.setItem(this.state.currentSite, JSON.stringify(newElements))
  }

  render() {
    const filteredItems = this.state.elements.filter((el) =>
      this.state.parent === undefined  ? el.parent === undefined : el.parent
&& this.state.parent.id === el.parent.id
    )
    const listItems = filteredItems.map((element) =>
      <ListBlock
        key={element.id}
        element={element}
        onBlockClick={this.handleBlockClick}
        onBlockDelete={this.handleBlockDelete}
        onBlockEdit={this.handleBlockEdit}
      />
    );
    return (
      <div className="App">
        {!this.state.showAdd &&
          <MainPage
            listItems={listItems}
            onAddClick={this.handleToggleAddClick}
            onBackClick={this.handleBackClick}
            onScrapClick={this.handleScrapClick}
            parent={this.state.parent}
            currentSite={this.state.currentSite}
          />
        }
        {this.state.showAdd &&
          <AddPage
            onAddElement={this.handleElementAdd}
            onCancelClick={this.handleToggleAddClick}
            editElement={this.state.editElement}
            parent={this.state.parent}
            currentSite={this.state.currentSite}
          />
        }
      </div>
    );
  }
}
```

```
export default App;
```

Файл App.css

```
.App, #root{
  height: 100%;
  min-height: 300px;
  min-width: 350px;
}

.page{
  display: flex;
  height: 100%;
  flex-direction: column;
  background: #FBFCFF;
  color: #222;
  font-family: "Montserrat", sans-serif;
  user-select: none;
}

h1, p{
  font-weight: 500;
}

.blue{
  background: #ECF4FF !important;
}

.red{
  background: #FBDFCC !important;
}

.green{
  background: #D5EFD6 !important;
}

.purple{
  background: #E7D9FC !important;
}

.grey{
  background: #EEEEEE !important;
}

select{
  width: 100%;
  border: none;
  border-radius: 10px;
  background: #F1F4FA;
  height: 40px;
  appearance: none;
  outline: none;
  color: #1F344A;
  text-indent: 14px;
  font-family: "Montserrat", sans-serif;
  font-size: 14px;
  cursor: pointer;
```

```
  font-weight: 500;
}

.select{
  position: relative;
}

select:disabled{
  color: #7F90A2;
  background: #ECEFF3;
  opacity: 1;
  cursor: default;
}

.select::after{
  content: "";
  position: absolute;
  width: 40px;
  height: 40px;
  background: no-repeat center url('./icons/arrow.svg');
  right: 0;
  bottom: 0;
  text-align: center;
  pointer-events: none;
}


.page .page-head{
  height: 70px;
  margin: 20px 20px 0px 20px;
}

.page h1{
  margin: 0px;
}

.page p{
  color: #9BA9B7;
  font-size: 14px;
  margin: 0;
}

.list{
  flex-grow: 1;
  overflow: auto;
  margin: 0px 20px 0px 20px;;
}

.list.add{
  height: 434px;
}

.list.empty{
  display: flex;
  justify-content: center;
  align-items: center;
```

```
}

.list .empty-list{
  text-align: center;
  display: none;
}

.empty-list img{
  animation: 5s infinite ease-in-out bounce;
}

.list.empty .empty-list{
  display: block;
}

.list .empty-list .text{
  font-size: 18px;
}

.page .button{
  display: flex;
  height: 50px;
  margin: 10px 20px 0px 20px;
  padding: 0;
  border: none;
  background: #ECF4FF;
  border-radius: 10px;
  cursor: pointer;
  transition: all .3s;
  z-index: 1;
}

.page .button:hover, .page .button:focus{
  outline: none;
}

.page .button:hover{
  background: #D9E9FF;
  transition: all .3s;
}

.page .button .icon{
  width: 50px;
  height: 50px;
  position: absolute;
}

.page .button .icon img{
  margin-top: 13px;
}

.page .button .text{
  font-size: 18px;
  color: #4793FF;
  text-align: center;
  font-weight: 500;
```

```
  flex-grow: 1;
  line-height: 50px;
}

.page .button.active{
  background: #3689FF;
  z-index: 2;
}

.button.bottom-margin{
  margin-bottom: 20px;
}

.page .button.active:hover{
  background: #3694FF;
  box-shadow: 0px 4px 20px rgba(54, 137, 255, 0.7);
}

.page .button.active .text{
  color: #FFFFFF;
}

.list-block{
  display: flex;
  margin-bottom: 6px;
  cursor: pointer;
  border-radius: 10px;
  transition: all .3s;
  position: relative;
  z-index: 1;
}

.list-block:hover .info .title{
  color: #3689FF;
  transition: all .3s;
}

.list-block .icon{
  width: 50px;
  height: 50px;
  border-radius: 10px;
  text-align: center;
}

.list-block .info{
  display: block;
  height: 50px;
  margin-left: 10px;
  border-bottom: 1px solid #F3F4F9;
  flex-grow: 1;
}

.list-block .info .title{
  padding-top: 3px;
  display: block;
  font-size: 18px;
```

```
  margin: 0;
  transition: all .3s;
}

.list-block .info .type{
  margin-top: 8px;
  font-size: 10px;
  text-transform: uppercase;
}

.list-block .icon img{
  margin-top: 13px;
}

.list-block .icon.block{
  background: #ECF4FF;
}

input{
  width: 100%;
  height: 40px;
  border: none;
  background: #F1F4FA;
  border-radius: 10px;
  color: #1F344A;
  text-indent: 14px;
  font-family: "Montserrat", sans-serif;
  font-weight: 500;
  font-size: 14px;
  outline: none;
  flex-grow: 1;
  padding: 0;
}

input:disabled{
  color: #7F90A2;
  background: #ECEFF3;
}

.error{
  background: #FBDFCC;
  border-radius: 10px;
  margin-top: 16px;
  padding: 14px;
}

.error p{
  color: #FF6902;
  font-size: 14px;
  font-weight: 500;
}

label{
  display: block;
  font-size: 16px;
  font-weight: 500;
```

```
  margin-bottom: 6px;
}

label:not(:first-child){
  margin-top: 12px;
}

.select-block{
  display: flex;
}

.select-block .button{
  width: 40px;
  height: 40px;
  margin: 0 0 0 6px;
}

.select-block .button .icon{
  width: 40px;
  height: 40px;
  position: relative;
}

.select-block .button .icon img{
  margin-top: 8px;
}

@keyframes bounce {
  0%, 100% {
    transform: translate3d(0,0,0);
  } 50%{
    transform: translate3d(0, -10px, 0);
  }30%, 70% {
    transform: translate3d(0, -20px, 0);
  }
}

.delete-button, .edit-button{
  position: absolute;
  margin-top: 5px;
}

.delete-button img, .edit-button img{
  padding: 4px;
}

.delete-button{
  right: 14px;
}

.edit-button{
  right: 50px
}

.delete-button:hover img, .edit-button:hover img{
  background: #F1F4FA;
```

```
  border-radius: 10px;
}

.button.disabled{
  filter: grayscale(1);
  cursor: not-allowed;
}
```

Файл content.js

```
/*global chrome*/
let tabID, resArr, dom, parent, firstElement, scrapArr;
let links = new Set()
let finded = new Set();

chrome.runtime.onMessage.addListener((request, sender, sendResponse) => {
    switch(request.message){
        case "start":
            parent = request.parent
            start(request.body);
            tabID = sender.id
            break;
        case "show":
            if(request.body.viewSelected){
                resArr = request.body.arr
                findElement(true)
            } else {
                clearSelected()
            }
            tabID = sender.id
            break;
        case "scrap":
            scrapArr = request.body
            scrap()
            break;
        default:
            break;
    }
    sendResponse({});
    return true;
});

function start(startSelect){
    if(startSelect){
        clearSelected()
        if(parent){
            firstElement = getFirstElement()
        }
        document.onmouseover = (event) => {
            event.stopPropagation();
            if(parent && !firstElement.contains(event.target)) return;
            event.target.classList.add('hint-mode_show-block');
        }
        document.onmouseout = (event) => {
            event.stopPropagation();
            if(parent && !firstElement.contains(event.target)) return;
            event.target.classList.remove('hint-mode_show-block');
```

```
        }
        document.onclick = (event) => {
            event.preventDefault();
            event.stopPropagation();
            if(parent && !firstElement.contains(event.target)) return;
            event.target.classList.remove('hint-mode_show-block');
            event.target.classList.toggle('selected-mode_show-block')
        }
    } else {
        const   selected   =   [...document.getElementsByClassName("selected-
mode_show-block")];
        if(selected.length > 0){
            resArr = getParams(selected[0]);
            for(let i = 1; i < selected.length; i++){
                for(let param in resArr){
                    if(resArr[param] !== getParams(selected[i])[param])delete
resArr[param]
                }
            }
        }
        chrome.runtime.sendMessage(tabID, {resArr: JSON.stringify(resArr)});
        clearSelected();
        document.onmouseover = undefined
        document.onmouseout = undefined
        document.onclick = undefined
        parent = undefined
    }
}

async function scrap(){
    dom = document.getElementsByTagName("*");
    const anchors =  scrapArr.filter(el => el.type === "anchor");
    const elements = scrapArr.filter(el => el.type !== "anchor");
    addLinks(dom);
    finded = new Set();
    const div = document.createElement('div');
    for(let link of links){
        console.log(link)
        await fetch(link)
        .then((response) => response.text())
        .then((html) => {
            div.innerHTML = html.trim();
            return div.getElementsByTagName("*");
        })
        // eslint-disable-next-line no-loop-func
        .then((newDOM) => {
            for(let anchor of anchors){
                if(!checkElement(newDOM, anchor)) return
            }
            addLinks(newDOM)
            for(let el of elements){
                resArr = el.selectArr
                findElement(false, newDOM)
            }
        })
    }
```

```
    let csvContent = "data:text/csv;charset=utf-8,";

    finded.forEach(function(rowArray) {
        console.log(rowArray)
        let row = rowArray.join(";");
        csvContent += row + "\r\n";
    });
    console.log(csvContent)
    var encodedUri = encodeURI(csvContent);
    var link = document.createElement("a");
    link.setAttribute("href", encodedUri);
    link.setAttribute("download", "my_data.csv");
    document.body.appendChild(link); // Required for FF
    link.click();
}

function addLinks(newDOM){
    for(let i = 0; i < newDOM.length; i++){
        if(newDOM[i].localName === "a"
        && newDOM[i].host === window.location.host
        && newDOM[i].href
        && newDOM[i]
        && !links.has(newDOM[i].href.split('#')[0])) {
            links.add(newDOM[i].href.split('#')[0])
        }
    }
}

function checkElement(dom, anchor){
    let flag = true;
    for(let i = 0; i < dom.length; i++){
        let elemParams = getParams(dom[i])
        flag = true
        for(let p in anchor.selectArr){
            if(anchor.selectArr[p] !== elemParams[p]){
                flag = false
                break
            }
        }
        if(flag) return true;
    }
    return false;
}

function findElement(show, dom){
    if(dom === undefined) dom = document.getElementsByTagName("*")
    let flag = true;
    for(let i = 0; i < dom.length; i++){
        let elemParams = getParams(dom[i])
        flag = true
        for(let p in resArr){
            if(resArr[p] !== elemParams[p]){
                flag = false
                break
            }
        }
```

```
        if(flag) {
            finded.add(dom[i])
            if(show) dom[i].classList.add('check-mode_show-block')
        }
    }
}

function clearSelected(){
    dom = document.body.getElementsByTagName("*");
    for(let i = 0; i < dom.length; i++){
        clearElementClass(dom[i])
    }
}

function clearElementClass(el){
    el.classList.remove('selected-mode_show-block')
    el.classList.remove('check-mode_show-block')
    el.classList.remove('check2-mode_show-block');
}

function getParams(el){
    clearElementClass(el)
    const params = {
        localName: el.localName,
        id: el.id,
        class: el.classList ? el.classList.value : "",
        childCount: el.children.length,
        parentNodeClass:   el.parentNode   &&   el.parentNode.classList   ?
el.parentNode.classList.value : undefined,
        parentNodeLocalName:   el.parentNode   ?   el.parentNode.localName   :
undefined,
        parentNodeId: el.parentNode ? el.parentNode.id : undefined,
        innerText:          el.innerText.slice(0,          20).replace("\n",
"").trim().replace(",", "").replace(";", "").replace("#", "")
    }
    return params
}

function getFirstElement(){
    let parArr = parent.selectArr
    dom = document.body.getElementsByTagName("*");
    let firstElement
    let flag = true
    for(let i = 0; i < dom.length; i++){
        let elemParams = getParams(dom[i])
        flag = true
        for(let p in parArr){
            if(parArr[p] !== elemParams[p]){
                flag = false
                break
            }
        }
        if(flag){
            firstElement = dom[i]
            dom[i].classList.add('check2-mode_show-block')
            break;
```

```
        }
    }
    return firstElement
}
```

Файл AddPage.js

```
import React from 'react';
/*global chrome*/
import crossImage from './icons/cross.svg';
import plusImage from './icons/plus.svg';
import cursorImage from './icons/cursor.svg';
import checkmarkImage from './icons/checkmark.svg';
import eyeImage from './icons/eye.svg';
import eyeoffImage from './icons/eyeoff.svg';
import trashImage from './icons/trash.svg';
import editwImage from './icons/edit-w.svg';

class AddPage extends React.Component{
      constructor(props){
            super(props);
            this.state={
                  newAdd: {
                        title: "",
                        type: "element",
                        selectArr: undefined,
                        parent: undefined
                  },
                  viewSelected: false,
                  selectingElements: false,
                  hasName: true,
                  hasSelector: true
            }

            this.handleSelectClick = this.handleSelectClick.bind(this);
            this.handleViewClick = this.handleViewClick.bind(this);
            this.handleDeleteClick = this.handleDeleteClick.bind(this);
            this.handleAddClick = this.handleAddClick.bind(this);
            this.handleCancelClick = this.handleCancelClick.bind(this);
            this.handleNameInputChange                                    =
this.handleNameInputChange.bind(this);
            this.handleSelectChange = this.handleSelectChange.bind(this);
      }

      componentDidMount() {
            const {editElement, parent} = this.props
            let    newAdd    =    editElement    ?    {...editElement}    :
{...this.state.newAdd}
            newAdd.parent = parent
            newAdd.type  =  editElement  ?  newAdd.type  :  parent  ?  "text"  :
"element";
            this.setState({
                  newAdd
            })
            chrome.runtime.onMessage.addListener((request,            sender,
sendResponse) => {
                  this.handleSelect(JSON.parse(request.resArr))
```

```
                sendResponse({});
                return true
            })
        }

        handleSelect(array){
            this.setState(prevState => ({
                newAdd: {
                    ...prevState.newAdd,
                    selectArr: array
                }
            }))
        }

        handleViewClick(){
            const viewSelected = !this.state.viewSelected
            const arr = this.state.newAdd.selectArr
            chrome.tabs.query({currentWindow: true, active: true}, function
(tabs){
                var activeTab = tabs[0];
                chrome.tabs.sendMessage(activeTab.id, {"message": "show",
"body": {viewSelected, arr}});
            });
            this.setState(prevState => ({
                viewSelected: !prevState.viewSelected,
                selectingElements: false
            }))
        }

        handleSelectClick(){
            const selectingElements = !this.state.selectingElements
            const parent = this.state.newAdd.parent
            chrome.tabs.query({currentWindow: true, active: true}, function
(tabs){
                var activeTab = tabs[0];
                chrome.tabs.sendMessage(activeTab.id, {"message": "start",
"body": selectingElements, "parent": parent});
            });
            this.setState(prevState => ({
                selectingElements: !prevState.selectingElements,
                viewSelected: false
            }))
        }

        handleDeleteClick(){
            chrome.tabs.query({currentWindow: true, active: true}, function
(tabs){
                var activeTab = tabs[0];
                chrome.tabs.sendMessage(activeTab.id, {"message": "show",
"body": {viewSelected: false}});
            });
            this.setState(prevState => ({
                newAdd: {
                    ...prevState.newAdd,
                    selectArr: undefined,
                },
```

```
                selectingElements: false,
                viewSelected: false
            }))
        }

        handleAddClick(){
            const {newAdd} = this.state
            let hasName = true
            let hasSelector = true
            if(!newAdd.title.length > 0) hasName = false
            if(!newAdd.selectArr) hasSelector = false
            chrome.tabs.query({currentWindow:  true,  active:  true},  function
(tabs){
                var activeTab = tabs[0];
                chrome.tabs.sendMessage(activeTab.id,   {"message":   "show",
"body": {viewSelected: false}});
            });
            this.setState({
                hasSelector,
                hasName
            })
            if(hasName && hasSelector)
                this.props.onAddElement(newAdd)
        }

        handleCancelClick(){
            this.props.onCancelClick()
        }

        handleNameInputChange(e) {
            const title = e.target.value
            this.setState(prevState => ({
                newAdd: {
                    ...prevState.newAdd,
                    title,
                },
                hasName: title.length > 0
            }))
        }

        handleSelectChange(e) {
            const type = e.target.value
            this.setState(prevState => ({
                newAdd: {
                    ...prevState.newAdd,
                    type
                }
            }))
        }

        render(){
            const   {newAdd,   selectingElements,   viewSelected,   hasName,
hasSelector, currentSite} = this.state
            const {editElement, parent} = this.props
            return(
                <div className="page">
```

```
                        <div class="page-head">
                              <h1 class="title">{editElement ? "Изменение
элемента" : "Новый элемент"}</h1>
                              <p class="description">{currentSite}</p>
                        </div>
                        <div class="list add">
                              <label>Название</label>
                              <input        type="text"        maxLength="12"
value={newAdd.title} onChange={this.handleNameInputChange}/>
                              <label>Выбор</label>
                              <div class="select-block">
                                    <input        type="text"        disabled
value={newAdd.selectArr    ?    newAdd.selectArr.localName    +    "."    +
newAdd.selectArr.class : "Пусто" }/>
                                    <button      class="button      green"
onClick={this.handleSelectClick}>
                                          <div class="icon">
                                                <img                    alt=""
src={selectingElements ? checkmarkImage : cursorImage}/>
                                          </div>
                                    </button>
                                    <button   class={newAdd.selectArr   ?
"button  blue  :  "button  blue  disabled"}  onClick={  newAdd.selectArr  ?
this.handleViewClick : ""}>
                                          <div class="icon">
                                                <img alt="" src={viewSelected ?
eyeoffImage : eyeImage}/>
                                          </div>
                                    </button>
                                    <button       class="button       red"
onClick={this.handleDeleteClick}>
                                          <div class="icon">
                                                <img alt="" src={trashImage}/>
                                          </div>
                                    </button>
                              </div>
                                    <label>Тип</label>
                                    <div class="select">
                                          {parent ?
                                          <select          value={newAdd.type}
onChange={this.handleSelectChange}>
                                                <option
value="text">Текст</option>
                                                <option
value="link">Ссылка</option>
                                                <option
value="image">Изображение</option>
                                          </select>
                                          :
                                          <select          value={newAdd.type}
onChange={this.handleSelectChange}>
                                                <option
value="element">Элемент</option>
                                                <option
value="anchor">Якорь</option>
                                          </select>
```

```
                                        }
                                    </div>
                                    {parent && (
                                        <React.Fragment>
                                            <label>Родитель</label>
                                            <input   type="text"   disabled
value={parent.title}/>
                                        </React.Fragment>
                                    )}
                                {(!hasName || !hasSelector) &&
                                    <div class="error">
                                        {!hasName && <p> - Введите название
элемента</p>}
                                        {!hasSelector   &&   <p>   -   Выберите
элемент на странице</p>}
                                    </div>
                                }
                            </div>
                            <button            className="button            active"
onClick={this.handleAddClick}>
                                <div className="icon">
                                    <img alt="" src={editElement ? editwImage :
plusImage}/>
                                </div>
                                <div className="text">{editElement ? "Изменить" :
"Добавить"}</div>
                            </button>
                            <button        className="button        bottom-margin"
onClick={this.handleCancelClick}>
                                <div className="icon">
                                    <img alt="" src={crossImage}/>
                                </div>
                                <div className="text">Отмена</div>
                            </button>
                        </div>
                    )
                }
}

export default AddPage;
```

Файл MainPage.js

```
import React from 'react';
import arrowsImage from './icons/arrows.svg';
import backImage from './icons/back.svg';
import plusImage from './icons/plus.svg';
import ghostImage from './icons/ghost.svg';

class MainPage extends React.Component{
  constructor(props){
    super(props);
    this.state={}
      this.handleAddClick = this.handleAddClick.bind(this);
      this.handleBackClick = this.handleBackClick.bind(this);
      this.handleScrapClick = this.handleScrapClick.bind(this);
  }
```

```
	handleAddClick() {
		this.props.onAddClick()
	}

	handleBackClick() {
		const {parent, onBackClick} = this.props
		onBackClick(parent && parent.parent ? parent.parent.id : undefined)
	}

	handleScrapClick() {
		this.props.onScrapClick()
	}

	render(){
		const {listItems, parent, currentSite} = this.props
		return(
			<div className="page">
				<div className="page-head">
					<h1 className="title">{parent ? "Схема \xAB" +
parent.title + "\xBB": "Элементы"}</h1>
					<p className="description">{currentSite}</p>
				</div>
				<div className={listItems.length > 0 ? "list" : "list
empty"}>
					{listItems.length > 0 && listItems}
					<div className="empty-list">
						<img alt="" src={ghostImage}/>
						<p className="text">Пусто</p>
					</div>
				</div>
				<button className="button active"
onClick={this.handleAddClick}>
					<div className="icon">
						<img alt="" src={plusImage}/>
					</div>
					<div className="text">Добавить элемент</div>
				</button>
				<button className="button bottom-margin"
onClick={parent ? this.handleBackClick : this.handleScrapClick}>
					<div className="icon">
						<img alt="" src={parent ? backImage :
arrowsImage}/>
					</div>
					<div className="text">{parent ? "Вернуться
назад" : "Собрать информацию"}</div>
				</button>
			</div>
		)
	}
}

export default MainPage;
```

Файл ListBlock.js

```
import React from 'react';
```

```
import blockImage from './icons/block.svg';
import textImage from './icons/text.svg';
import linkImage from './icons/link.svg';
import picImage from './icons/pic.svg';
import anchorImage from './icons/anchor.svg';
import deleteButton from './icons/delete.svg';
import editButton from './icons/edit.svg';


class ListBlock extends React.Component{
      constructor(props){
            super(props);
            this.state={}
            this.handleDeleteClick = this.handleDeleteClick.bind(this)
            this.handleEditClick = this.handleEditClick.bind(this)
            this.handleListBlockClick = this.handleListBlockClick.bind(this)
      }

      handleEditClick(e){
            const {onBlockEdit, element} = this.props
            e.stopPropagation();
            onBlockEdit(element)
      }

      handleDeleteClick(e){
            const {onBlockDelete, element} = this.props
            e.stopPropagation();
            onBlockDelete(element.id)
      }

      handleListBlockClick(){
            const {onBlockClick, element} = this.props
            onBlockClick(element)
      }

      render(){
            let {element} = this.props
            const icons = {
                  "element": ["blue", blockImage, "Элемент"],
                  "text": ["red", textImage, "Текст"],
                  "link": ["green", linkImage, "Ссылка"],
                  "image": ["purple", picImage, "Изображение"],
                  "anchor": ["grey", anchorImage, "Якорь"]
            }
            return (
                  <div   className="list-block"   onClick={element.parent   ||
element.type === "anchor" ? this.handleEditClick : this.handleListBlockClick}>
                        <div       className={"icon       block       "       +
icons[element.type][0]}>
                              <img alt="" src={icons[element.type][1]}/>
                        </div>
                        <div className="info">
                              <h1 className="title">{element.title}</h1>
                              <p className="type">{icons[element.type][2]}</p>
                        </div>
```

```
                        {element.parent  ===  undefined  &&  element.type  !==
"anchor" &&
                              <div                      className="edit-button"
onClick={this.handleEditClick}>
                                    <img alt="" src={editButton}/>
                              </div>
                        }
                        <div                        className="delete-button"
onClick={this.handleDeleteClick}>
                              <img alt="" src={deleteButton}/>
                        </div>
                  </div>
            )
      }
}

export default ListBlock;
```