УДК 004.4, 004.9

# АВТОМАТИЧЕСКОЕ АННОТИРОВАНИЕ HTML-ДОКУМЕНТОВ ПО СТАНДАРТУ MICRODATA

**Т. Ф. Ибрагимов[1] [0009-0008-4959-5259], А. А. Ференец[2] [0000-0002-7859-9901]**

*[1, 2]Институт информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, ул. Кремлевская, 35, г. Казань, 420008*

[1]i.timur0701@gmail.com, [2]ist.kazan@gmail.com

***Аннотация***

Описана разработка на основе методов машинного обучения приложения для автоматического аннотирования веб-страниц по стандарту Microdata с возможностью расширения для других стандартов и с внедрением данных в JSX-файлы. Собраны и подготовлены датасеты для обучения моделей Machine Learning (ML). Собраны и проанализированы метрики модели ML.

***Ключевые слова**: Microdata, семантическая разметка, HTML5, поисковая оптимизация (SEO), поисковые системы, машинное обучение, schema.org, семантический веб, стандарты разметки, автоматизация SEO.*

## ВВЕДЕНИЕ

Современным стандартом для разметки веб-страниц является HTML5 (HyperText Markup Language) [1]. Хотя HTML-элементы предоставляют полезную дополнительную информацию о разделах документа, они описывают лишь общие понятия: навигация, заголовок, ссылка и подобные. Однако в HTML5 отсутствует конкретизация для таких элементов, как фильмы, товары и другие. Этот недостаток в языке разметки преодолевается с помощью описания элементов согласно стандарту «Микроданные», который позволяет встраивать структурированные метаданные в веб-страницы. Авторы вставляют машиночитаемую информацию в свои HTML веб-страницы, которая помогает определить, какой текст является названием фильма, какой – именем режиссера и т. д.

Семантически аннотированные данные на веб-сайте улучшают доступность информации при анализе ботами, в том числе поисковыми, лучше индексируются

и поднимаются в поисковой выдаче. Различные исследования показывают, что все больше и больше ресурсов внедряют семантическую разметку, потому что такая оптимизация (далее SEO) может значительно увеличить пользовательский трафик ресурса.

Семантическую разметку можно внедрить на всех ресурсах, использующих HTML-страницы для представления информации в интернете. Однако сейчас разработчики предпочитают использовать различные технологии, упрощающие процесс создания веб-страниц, которые в свою очередь формируют HTML. Среди таких технологий можно выделить различные шаблонизаторы, которые описывают шаблоны формирования динамических страниц, или в том числе синтаксис JSX [2], который был принят для использования множеством фреймворков. Разнообразие инструментария, используемого при разработке, вносит дополнительную сложность в процесс аннотирования. При этом процесс аннотирования может дополнительно усложняться для больших веб-сайтов из-за роста количества и объема страниц. Поэтому разработка инструмента, автоматизирующего процесс внедрения семантической разметки, важна для обеспечения простоты и скорости разработки веб-приложений.

Существует несколько способов внедрить семантическую разметку: с помощью стандартов Microdata [3] и JSON-LD [4]. Последнее исследование [5] показало, что применение JSON-LD более распространено, однако в данной работе рассматривается только Microdata, потому что не все популярные поисковые системы на данный момент поддерживают формат JSON-LD.

**Обзор предметной области**

Семантическая разметка используется для аннотирования различных типов данных, таких как товары, события, статьи, организации, фильмы и другие. Например, веб-страница, описывающая фильм, может содержать семантические метаданные, указывающие название фильма, его режиссера, год производства и рейтинг.

Все классы и атрибуты онтологии инициативы Schema.org [6] используются совместно с одним из форматов, с помощью которого эти метаданные внедряются в страницу. Среди используемых форматов: Microdata, JSON-LD, RDFa [7],

Microformats [8]. Наиболее популярными форматами сейчас являются Microdata и JSON-LD.

Для разработки инструмента семантической разметки были определены следующие ключевые критерии:

- Необходима поддержка стандарта Microdata.
- Инструмент должен автоматически добавлять аннотации в HTML-код.
- Инструмент должен иметь возможность расширения набора поддерживаемых понятий из онтологии Schema.org.
- Инструмент должен позволять переносить аннотации в исходный код шаблонов представления данных.

Нами были проведены поиск и анализ существующих решений. В результате установлено, что:

- Microdata Generator [9] предоставляет возможность создания JSON-LD-объекта или HTML-фрагмента для внедрения в веб-страницу. Процесс обработки данных не автоматизирован.
- Отличительной особенностью мастера разметки структурированных данных Google [10] является возможность аннотировать элементы страницы в интерактивном режиме. Этот инструмент предоставляет два способа, с помощью которых можно аннотировать страницу: использовав ссылку на веб-страницу либо передав HTML, который необходимо аннотировать.
- InLinks [11], в отличие от предыдущих инструментов, автоматически определяет тематику страницы на основе ее содержимого и генерирует объекты JSON-LD для внедрения. Однако он не поддерживает стандарт Microdata, и все созданные объекты имеют тип Thing.
- Web-segment [12] также автоматизирует процесс обработки данных с поддержкой стандарта Microdata, но поддерживает всего лишь 1 класс из онтологии Schema.org.

Таким образом, в большинстве инструментов нет возможности обработки данных в автоматическом режиме, в некоторых отсутствует поддержка достаточного количество классов из онтологии Schema.org, а InLinks вовсе не поддерживает стандарт Microdata.

**Проектирование программного решения**

Внедрению семантической разметки предшествует несколько этапов: определение страниц, которые необходимо аннотировать, определение типов объектов, находящихся на странице, и выбор соответствующих схем из онтологии Schema.org. В данной работе предложен подход, основанный на машинном обучении для определения типов объектов, выбора соответствующих схем и последующего внедрения семантической разметки на страницу в автоматическом режиме.

Для генерации семантической разметки необходимо получить HTML-документ, для которого будет сгенерирована семантическая разметка. Затем, использовав полученный документ и найденные типы объектов на странице, необходимо выбрать соответствующие схемы объектов из онтологии Schema.org. При необходимости, для переноса данных из сгенерированного HTML-документа в файлы, использующие технологии создания веб-страниц, отличные от HTML, мы предполагаем использование независимого инструмента. Такая модульность позволит в будущем, при необходимости поддержки той или иной технологии создания веб-страниц, разработать отдельный механизм, учитывая специфику соответствующей технологии.

Концепция работы инструмента предполагает последовательное выполнение нескольких этапов обработки данных.

1. Изначально необходимо получить документ, в который предполагается внедрить семантическую разметку. Документ можно получить путем передачи URL-адреса страницы в интернете, чтобы она впоследствии была загружена и обработана, либо передав локальный путь доступа к документу.

2. На следующем этапе происходит предобработка документа. Страницу необходимо очистить от элементов, которые не несут семантически значимую информацию. Такие элементы впоследствии могут замедлить выполнение программы и ухудшить результаты работы всего алгоритма, именно поэтому на этапе предобработки документа эти элементы должны быть удалены.

3. На следующем этапе блоки данных сегментируются для их последующей классификации путем поиска в документе шаблонов. При генерации данных на серверной или клиентской частях используются подготовленные шаблоны. Полученная информация из базы данных или с сервера применяется к подготовленным

шаблонам, группируется, и в результате генерируется HTML-страница. Поэтому, использовав алгоритм, основанный на сопоставлении структурно схожих элементов, можно сегментировать эти блоки данных для их последующей классификации.

4. На этапе классификации необходимо определить классы объектов онтологии Schema.org, найденных на предыдущем шаге, и сгенерировать css-селекторы для получения элементов.

5. На следующем этапе на основе css-селекторов, сгенерированных на предыдущем шаге, генерируется разметка Microdata. Каждый класс онтологии определяет соответствующий набор атрибутов класса. Для выявленного класса с помощью регулярных выражений и по определенным правилам генерируются соответствующие атрибуты.

6. После создания разметки алгоритм прекращает свою работу и генерирует HTML-документ.

7. С использованием полученных данных, семантическая разметка должна быть внедрена в соответствующие элементы в файле с синтаксисом JSX.

**Реализация инструмента**

Задача предобработки документа решается нами методами машинного обучения с помощью бинарной классификации. Таким образом, один из классов представляет собой элемент, несущий семантическую информацию, которая впоследствии может быть использована при обработке данных, а другой класс – декоративный элемент либо элемент, не несущий полезной информации, он может быть удален из документа без последствий для работы алгоритма.

При разработке модели машинного обучения было использовано консольное приложение, разработанное Nichita Utiu, Vlad-Sebastian Ionescu [13].

Модель была обучена на данных, извлеченных из датасета Dragnet [14]. Он состоят из HTML-страниц, которые были собраны Matthew E. Peters, Dan Lecocq из следующих источников:

- 999 страниц было выбрано через RSS-каналы из ресурсов с большим количеством подписчиков;
- 204 страницы были собраны из 23 популярных новостных ресурсов;
- 178 страниц были случайно выбраны из списка различных блогов.

Каждый тег в датасете представлен следующим набором признаков:

- глубина вложенности;
- позиция среди соседних элементов;
- количество дочерних узлов;
- длина текста;
- длина атрибута класса;
- длина атрибута id;
- категориальный признак, содержащий тип тега.

Также в набор данных включены признаки элементов-предков и дочерних элементов.

Для элементов-предков использованы те же характеристики, что и для рассматриваемого тега: извлечена информация для 5 элементов, являющихся предками тега. Различные исследования (например, [15, 16]) показали, что путь от корня до элемента может нести полезную информацию и положительно влияет на работу подобного решения. Для дочерних элементов упомянутые выше характеристики усреднялись и группировались вместе. Эти усредненные значения вместе с уровнем глубины вложенности извлекались для 5 уровней вложенности в поддереве рассматриваемого элемента. Если этих данных недостаточно, то в случае элементов-предков они заполнялись значениями самого верхнего элемента-предка, а в случае дочерних элементов набор заполнялся значениями самого нижнего уровня.

Для задачи классификации была использована модель, основанная на алгоритме случайного леса, поскольку эта модель, исходя из заключений в ранее упомянутой статье [13], показывает себя наилучшим образом.

Основной задачей алгоритма сегментации является поиск повторяющихся шаблонов для оптимизации последующей классификации. Когда пользователь заходит на веб-сайт в браузере, серверный или клиентский скрипт вставляет результаты, полученные из базы данных, в заранее определенный фрагмент шаблона HTML-кода. Затем эти данные группируются и встраиваются в HTML-документ. Использовав алгоритм, основанный на сопоставлении блоков страницы и поиске структурно схожих элементов, можно сегментировать эти блоки данных.

Например, в онлайн-магазинах часто можно увидеть карточки товаров, которые содержат изображение, название товара и цену. Эти элементы имеют схожую DOM-структуру, поэтому, сопоставив эти элементы, можно выделить эти блоки данных.

Сегментация происходит в несколько этапов:

1. На первом этапе происходит обход дерева в ширину, где каждый элемент получает два атрибута: уникальный идентификатор и количество дочерних элементов родителя элемента;
2. Следующий этап – сравнение элементов дерева. На этом этапе генерируется значение динамического окна от 1 до максимального значения, равного половине количества дочерних элементов родителя. Выбор такого максимального значения объясняется тем, что как минимум 2 элемента могут быть сопоставлены. Затем выбирается опорный элемент, на основе которого заполняются окна. Первое окно содержит элементы, идущие перед опорным, второе – элементы, включая опорный элемент и после него, третье – элементы, идущие после опорного. Затем структуры элементов второго и первого, второго и третьего окон сопоставляются путем приведения последовательно идущих тегов к строке вида tagAtagB. Если структура элементов второго окна совпала со структурой элементов другого окна, то элементы второго окна выводятся в качестве блоков.
3. На последнем этапе происходит генерация css-селекторов блоков, которые были получены на предыдущем шаге. Рекурсивно до тега html включительно для каждого элемента генерируется строка вида nodeName:nth-child(nodePosition), где nodeName – тип тега, nodePosition – позиция среди соседних элементов, либо строка вида nodeName, где nodeName – название тега, для случаев, когда у узла нет соседних элементов. Затем эти строки конкатенируются между собой с помощью символа “>”, указывающего на вложенность последующего элемента в предыдущий.

Для обучения модели обработки естественного языка было необходимо подготовить данные, содержащие семантически размеченные текстовые данные. В качестве источника был использован ресурс, подготовленный в рамках проекта Web Data Commons [17] и содержащий извлеченные семантически размеченные сущности из корпуса Common Crawl 2022 [18].

Для извлечения данных, хранящихся в формате n-quad rdf, был использован язык SPARQL [19]. Для получения необходимых данных были составлены запросы для извлечения выделенных сущностей: Event, Book, Movie, Product, Hotel, Restaurant, JobPosting, Recipe.

Для каждой сущности было извлечены по 10 000 записей. Такой объем выбран с целью обеспечить достаточное разнообразие элементов для последующего обучения модели обработки естественного языка.

Для обработки естественного языка была выбрана мультиязычная BERT [20]. Для обучения модели был использован набор данных объемом 70 000 объектов онтологии Schema.org, извлеченный ранее из корпуса Common Crawl 2022 [17]. Набор данных был разделен на обучающий и валидационный наборы данных в соотношении 80% и 20% соответственно.

После этапа обучения был произведен замер показателей производительности модели (рис. 1). Для оценки качества модели работы были использованы следующие известные метрики: точность (precision), полнота (recall) и F-мера (f1-score).

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Product | 0.94 | 0.92 | 0.93 | 1001 |
| Book | 0.81 | 0.70 | 0.75 | 981 |
| Event | 0.84 | 0.88 | 0.86 | 962 |
| Hotel | 0.95 | 0.95 | 0.95 | 1015 |
| JobPosting | 1.00 | 0.98 | 0.99 | 972 |
| Movie | 0.73 | 0.83 | 0.78 | 1048 |
| Recipe | 0.94 | 0.89 | 0.91 | 1000 |
| Restaurant | 0.83 | 0.87 | 0.85 | 1021 |
| accuracy | | | 0.88 | 8000 |
| macro avg | 0.88 | 0.88 | 0.88 | 8000 |
| weighted avg | 0.88 | 0.88 | 0.88 | 8000 |

Рис. 1. Значения метрик для оценки производительности и качества предсказаний модели

Также для более детального анализа качества модели была построена матрица ошибок (confusion matrix). На пересечении строки и столбца с одним классом указан процент правильно идентифицированных объектов, а в прочих ячейках указаны проценты ошибочных классификаций (рис. 2).

Из полученной матрицы видно, что модель часто ошибочно идентифицирует класс Book как класс Movie и, наоборот, класс Movie как класс Book. Скорее всего, это связано с семантической близостью классов этих объектов. Часто могут встречаться фильмы и книги с одинаковыми или похожими названиями, при этом модель не учитывает контекст появления этих объектов и опирается только на название и описание объектов, из-за чего модель может ошибочно классифицировать эти объекты.

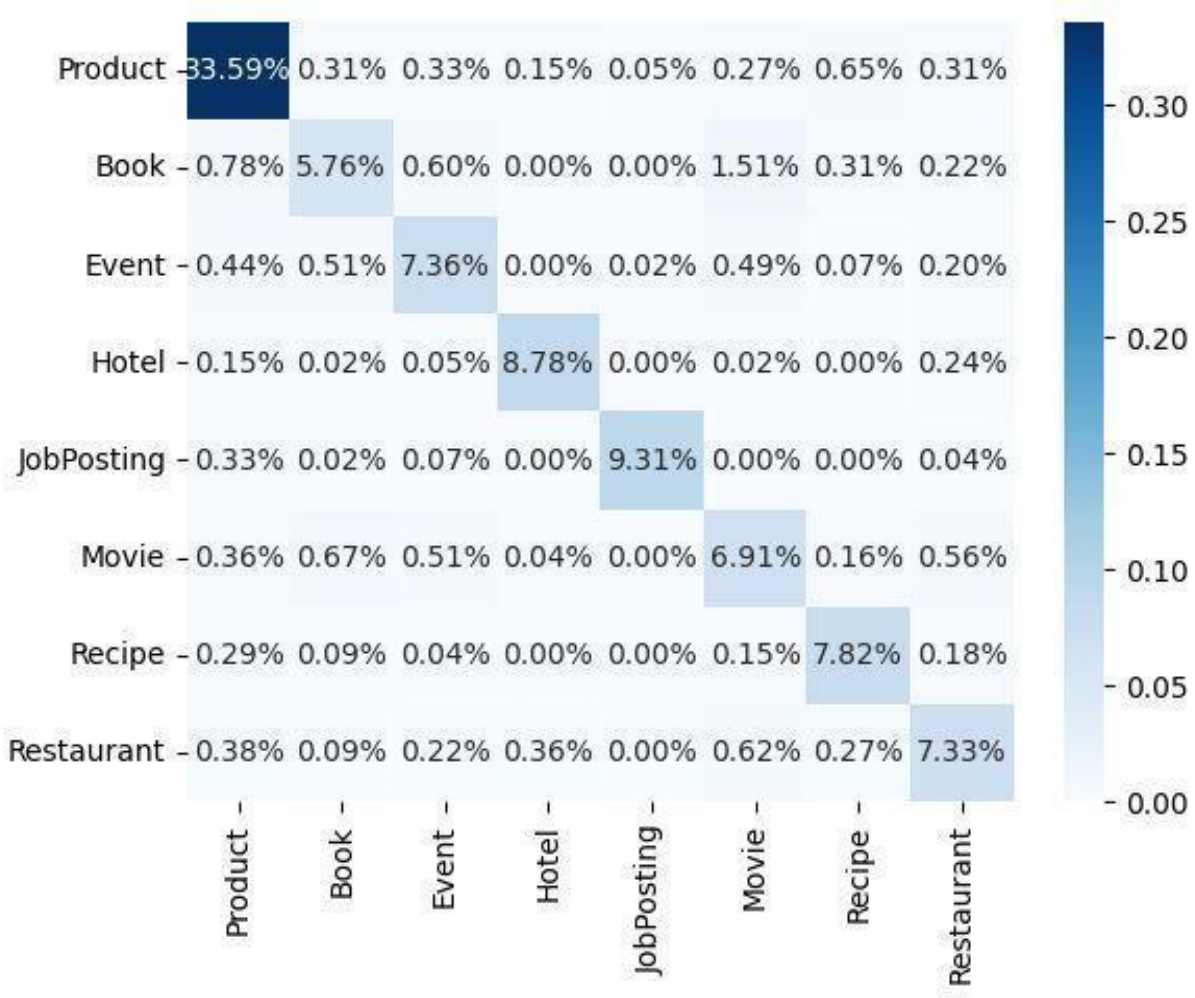


Рис. 2. Матрица ошибок (confusion matrix) модели

Существует большая вариативность выбора технологий для создания веб-страниц, включая различные шаблонизаторы, синтаксис JSX и т. д. Несмотря на то, что возможности этих технологий во многом совпадают, чтобы перенести данные из страницы с автоматически сгенерированной семантической разметкой, используя ту или иную технологию, необходимо учитывать синтаксис, специфический для этой технологии. Поэтому этап автоматического аннотирования был разделен на два независимых механизма: генерация семантической разметки страницы и перенос данных, реализация которого может отличаться для различных технологий. Поскольку JSX уже стал стандартом в современной разработке интерфейсов, эта технология была выбрана для реализации переноса данных.

Для переноса семантической разметки из исходной страницы в JSX сначала необходимо построить абстрактное синтаксическое дерево (AST-дерево) и сопоставить его с исходной страницей, и в случае, если они совпадают, аннотировать соответствующие узлы в JSX.

В качестве парсера для получения AST-дерева были использованы компилятор Babel [21] и язык TypeScript [22]. Обязательным условием для парсинга является наличие экспорта по умолчанию для функции, содержащей JSX. В AST-дереве, помимо обычных тегов, могут встречаться выражения, отличные от объявления тегов, например, объявление переменной. Но поскольку HTML-дерево не содержит этих объявлений, необходимо построить преобразованное дерево, полностью копирующее структуру AST-дерева без выражений, отличных от объявлений HTML-элементов. Для этого была использована рекурсивная функция, которая на вход получает текущий узел и массив, агрегирующий дочерние элементы, в который необходимо передать AST-узел с названием тега, номером строки, на которой появляется тег, номером первого символа в строке и набором семантических атрибутов. Таким образом, на выходе получается структура, полностью копирующая HTML-дерево. Если структуры деревьев совпадут, то происходит генерация групп, содержащих семантическую разметку, которую необходимо внедрить. Группа характеризуется местоположением для внедрения семантических атрибутов и набором связанных с этим узлом семантических атрибутов. Для генерации групп производится последовательный обход HTML-дерева. Если при обходе узла встречаются один или несколько семантических атрибутов, то в

качестве номера строки сохраняется строка, в которой находится этот узел в исходном файле, а в качестве номера символа в строке указывается номер символ в строке, находящийся перед закрывающим символом тега “>” На следующем этапе происходит внедрение разметки на основе групп, сгенерированных на предыдущем этапе. По номеру строки и номеру символа в строке происходит вставка подготовленный строки, содержащей необходимую разметку.

**ЗАКЛЮЧЕНИЕ**

Результатом работы является программное приложение для автоматического аннотирования по стандарту Microdata, построенное с применением технологий машинного обучения. Им могут пользоваться авторы веб-страниц для улучшения позиций при ранжировании страниц поисковыми системами. Приложение можно доработать, добавив поддержку формата JSON-LD и расширив список технологий, позволяющих его использовать.

**СПИСОК ЛИТЕРАТУРЫ**

1. HTML5 (HyperText Markup Language).
URL: https://html.spec.whatwg.org/multipage/introduction.html.
2. JSX. URL: https://www.typescriptlang.org/docs/handbook/jsx.html.
3. Microdata.
URL: https://html.spec.whatwg.org/multipage/microdata.html.
4. JSON-LD. URL: https://json-ld.org.
5. *Brinkmann A., Primpeli A., Bizer Ch.* The Web Data Commons Schema.org Data Set Series.
URL: https://www.uni-mannheim.de/media/Einrichtungen/dws/Files_Research/Web-based_Systems/pub/Brinkmann-etal-TheWDCSchemaorgDataSetSeries-WWW2023.pdf.
6. Schemas. URL: https://schema.org/docs/schemas.html.
7. RDFa. URL: https://www.w3.org/TR/html-rdfa.
8. Microformats. URL: https://microformats.org.
9. Local Business Schema Generator – MicroData & JSON-LD.
URL: https://microdatagenerator.org/localbusiness-microdata-generator.

10. Structured Data Markup Helper. URL: https://www.google.com/webmasters/markup-helper/u/0.

11. Entity SEO Tools. URL: https://inlinks.com/.

12. Web-segment. URL: https://github.com/liaocyintl/web-segment.

13. *Utiu N., Ionescu V.-S.* Learning Web Content Extraction with DOM Features. URL: http://dx.doi.org/10.1109/ICCP.2018.8516632.

14. *Peters M. E., Lecocq D.* Content extraction using diverse feature sets // WWW '13 Companion: Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, May 13–17, 2013. Association for Computing Machinery, New York, NY, United States: 2013, pages 89–90.

15. *Gongqing Wu, Li Li, Xuegang Hu, Xindong Wu* Web news extraction via path ratios. URL: https://dl.acm.org/doi/abs/10.1145/2505515.2505558.

16. *Vadrevu S., Gelgi F., Davulcu H*. Semantic partitioning of web pages // Web Information Systems Engineering–WISE 2005: 6th International Conference on Web Information Systems Engineering, New York, NY, USA, November 20–22, 2005. Proceedings 6. – Springer Berlin Heidelberg, 2005. P. 107–118.

17. Extraction Results from the October 2022 Common Crawl Corpus. URL: https://webdatacommons.org/structureddata/#results-2022-1.

18. Common Crawl September/October 2022 Crawl Archive (CC-MAIN-2022-40). URL: https://data.commoncrawl.org/crawl-data/CC-MAIN-2022-40/index.html.

19. SPARQL Query Language. URL: https://www.w3.org/TR/sparql11-query.

20. BERT: https://arxiv.org/abs/1810.04805.

21. Babel. URL: https://babeljs.io/.

22. TypeScript. URL: https://www.typescriptlang.org/.

# AUTOMATIC ANNOTATION OF HTML DOCUMENTS USING THE MICRO-DATA STANDARD

**T. F. Ibragimov[1] [0009-0008-4959-5259], A. A. Ferenets[2] [0000-0002-7859-9901]**

[1, 2]Institute of Information Technology and Intelligent Systems, Kazan Federal University

[1]i.timur0701@gmail.com, [2]ist.kazan@gmail.com

***Abstract***

The development of an application based on machine learning methods for automatic annotation of web pages according to the Microdata standard is described, with the possibility of extension to other standards and injecting data to JSX files. Datasets were collected and prepared for training Machine Learning (ML) models. The ML model metrics were collected and analyzed.

***Keywords****: Microdata, semantic markup, HTML5, search engine optimization (SEO), search engines, machine learning, schema.org, semantic web, markup standards, SEO automation.*

**REFERENCES**

1. HTML5 (HyperText Markup Language). URL: https://html.spec.whatwg.org/multipage/introduction.html.
2. JSX. URL: https://www.typescriptlang.org/docs/handbook/jsx.html.
3. Microdata. URL: https://html.spec.whatwg.org/multipage/microdata.html.
4. JSON-LD. URL: https://json-ld.org.
5. *Brinkmann A., Primpeli A., Bizer Ch.* The Web Data Commons Schema.org Data Set Series. URL: https://www.uni-mannheim.de/media/Einrichtungen/dws/Files_Research/Web-based_Systems/pub/Brinkmann-etal-TheWDCSchemaorgDataSetSeries-WWW2023.pdf.
6. Schemas. URL: https://schema.org/docs/schemas.html.
7. RDFa. URL: https://www.w3.org/TR/html-rdfa.
8. Microformats. URL: https://microformats.org.

9. Local Business Schema Generator – MicroData & JSON-LD. URL: https://microdatagenerator.org/localbusiness-microdata-generator.

10. Structured Data Markup Helper. URL: https://www.google.com/webmasters/markup-helper/u/0.

11. Entity SEO Tools. URL: https://inlinks.com/.

12. Web-segment. URL: https://github.com/liaocyintl/web-segment.

13. *Utiu N., Ionescu V.-S.* Learning Web Content Extraction with DOM Features. URL: http://dx.doi.org/10.1109/ICCP.2018.8516632.

14. *Peters M. E., Lecocq D.* Content extraction using diverse feature sets // WWW '13 Companion: Proceedings of the 22nd International Conference on World Wide Web, Rio de Janeiro, Brazil, May 13–17, 2013. Association for Computing Machinery, New York, NY, United States: 2013, pages 89–90.

15. *Gongqing Wu, Li Li, Xuegang Hu, Xindong Wu* Web news extraction via path ratios. URL: https://dl.acm.org/doi/abs/10.1145/2505515.2505558.

16. *Vadrevu S., Gelgi F., Davulcu H*. Semantic partitioning of web pages // Web Information Systems Engineering–WISE 2005: 6th International Conference on Web Information Systems Engineering, New York, NY, USA, November 20–22, 2005. Proceedings 6. – Springer Berlin Heidelberg, 2005. P. 107–118.

17. Extraction Results from the October 2022 Common Crawl Corpus. URL: https://webdatacommons.org/structureddata/#results-2022-1.

18. Common Crawl September/October 2022 Crawl Archive (CC-MAIN-2022-40). URL: https://data.commoncrawl.org/crawl-data/CC-MAIN-2022-40/index.html.

19. SPARQL Query Language. URL: https://www.w3.org/TR/sparql11-query.

20. BERT: https://arxiv.org/abs/1810.04805.

21. Babel. URL: https://babeljs.io/.

22. TypeScript. URL: https://www.typescriptlang.org/.

## СВЕДЕНИЯ ОБ АВТОРАХ

***ИБРАГИМОВ Тимур Фердинандович*** – выпускник бакалавриата Института информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, г. Казань.

***Timur Ferdinandovich IBRAGIMOV*** – graduate student of the Institute of Information Technologies and Intelligent Systems, Kazan (Volga region) Federal University, Kazan.

Email: i.timur0701@gmail.com

ORCID: 0009-0008-4959-5259

***ФЕРЕНЕЦ Александр Андреевич*** – старший преподаватель кафедры программной инженерии Института информационных технологий и интеллектуальных систем Казанского (Приволжского) федерального университета, г. Казань.

***Alexander Andreevich FERENETS*** – senior lecturer of Software Engineering of Institute of Information Technologies and Intelligent Systems KFU.

Email: ist.kazan@gmail.com

ORCID: 0000-0002-7859-9901

*Материал поступил в редакцию 20 июля 2024 года*