МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Кафедра 319 «Интеллектуальные системы мониторинга»

КУРСОВАЯ РАБОТА

по дисциплине
«Основы построения промышленных программных систем»

«Проектирование и разработка веб-приложения классификации текстов с применением методов машинного обучения»

Студент Мицич А.Д.

Группа М3О-221М-20

Руководитель _____ Полицына Е. В.

Оценка _____ Дата защиты «25» декабря 2021 г.

Москва 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(национальный исследовательский университет)»

Кафедра 319 «Интеллектуальные системы мониторинга»

УТВЕРЖДАЮ

Заведующий кафедрой
д.т.н., профессор Нагибин С.Я.
(И.О.Фамилия)
«____»_____ 2021 г.

З А Д А Н И Е

на курсовую работу по дисциплине
Основы построения промышленных программных систем
Студент М3О-221М-20, Мицич Андрей Драгишаевич
(№ группы, Ф. И. О.)

Тема «Проектирование и разработка веб-приложения классификации текстов с применением методов машинного обучения»

Перечень вопросов, подлежащих разработке в курсовой работе

Реализовать клиент-серверную систему классификации для свой предметной области, которая представляет собой веб-приложение на Java/Python с пользовательским веб- интерфейсом.

Рекомендуемая литература

1. Django. Разработка веб-приложений на Python | Форсье Джефф, Биссекс Пол
2. Django. Адриан Головатый, Джейкоб Каплан-Мосс
3. «Machine Learning, Neural and Statistical Classification» D. Michie, D.J. Spiegelhalter, C.C. Taylor.

Задание выдано «12» сентября 2021 г.
Руководитель Полицына Е.В., к.т.н. доцент кафедры 319_____
(Ф. И. О., должность, подпись)

Студент_____
(подпись)

СОДЕРЖАНИЕ

1 ОПИСАНИЕ ВОЗМОЖНОСТЕЙ ПРИЛОЖЕНИЯ

1.1 Общие сведения

1.1.1 Полное наименование системы и её условное обозначение

Веб-приложение «TextClassifierWebService».

Условное обозначение: Классификатор текста.

1.1.2 Наименование предприятий разработчика и заказчика системы

Заказчик – кафедра 319, МАИ (национальный исследовательский университет), Полицына Е. В.

Разработчик – студент группы М3О-221М-20 Мицич А.Д.

1.1.3 Основания для разработки

Курсовая работа по предмету «Основы построения промышленных программных систем», целью которой является изучение методов машинного обучения, проектирование, моделирование, разработка и тестирование веб-приложения с использованием изученных методов классификации.

1.1.4 Плановые сроки начала и окончания работы по созданию системы

Начало работ по проектированию и реализации системы: сентябрь 2021 г.

Окончание работ по созданию системы: декабрь 2021 г

1.1.5 Порядок оформления и предъявления заказчику результатов работ по созданию системы

К результатам труда разработчика относится: программное обеспечение, документация.

Заказчику передаются:

- ссылка на репозиторий GitHab с реализованным программным обеспечением;
- архив формата .rar с реализованным программным продуктом и сопроводительной документации проекта;
- подшитая сопроводительная документация проекта.

1.2 Назначения и цели

1.2.1 Назначение системы

Классификатор предназначен для классификации текста по категориям. На данный момент система классифицирует текст по 5 категориям:

- экономика;
- наука и техника;
- интернет и СМИ;
- спорт;
- культура.

1.2.2 Цели создания системы

Целью создания системы является:

- автоматизация классификации текста;
- анализ результатов работы классификатора;
- автоматизация процессов хранения, обработки, добавления и удаления текстов.

2 ТРЕБОВАНИЯ

2.1 Требования к системе

2.1.1 Требования к структуре и функционированию системы

Система должна иметь клиент-серверную архитектуру. Пользовательский интерфейс должен быть реализован в вебе. В качестве протокола взаимодействия между клиентом и сервером следует использовать протокол прикладного уровня – HTTP.

2.1.2 Требования к эргономике и технической эстетике

1. Наличие русскоязычного пользовательского веб – интерфейса.

2. Веб-приложение должно иметь дружелюбный интерфейс.

3. При невозможности выполнения какого – либо действия должно выводиться диагностическое сообщение.

2.1.3 Требования по стандартизации и унификации

Для разработки серверной части приложения должен использоваться язык Python не ниже версии 3.0. Для реализации классификатора могут быть использованы любые библиотеки, реализующие методы машинного обучения. Для реализации векторизации текстов можно использовать любые библиотеки, фреймворки, API и т.д.

Для разработки клиентской части приложения могут использоваться любые технологии.

2.2 Функциональные требования

- создание статей;
- просмотр статей;
- удаление статей;
- обновление статей;
- фильтрация и сортировка статей по признакам, который укажет пользователь;
- классификация текста с применением технологии машинного обучения.

2.3 Нефункциональные требования

1. Соответствуют требованиям к эргономике и технической эстетике (п. 2.1.2).

3 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

Для разработки веб-приложения был выбран фреймворк Django на языке Python(с подробными версиями технологий можно ознакомиться ниже).

В качестве основного шаблона проектирования архитектуры приложения был выбран паттерн MVT (Model-View-Template). Вся логика при таком подходе вынесена во View, а то, как будут отображаться данные в Template. Из-за ограничений HTTP протокола, View в Django описывает, какие данные будут представленны по запросу на определенный URL. View, как и протокол HTTP, не хранит состояний и по факту является обычной функцией обратного вызова, которая запускается вновь при каждом запросе по URL. Шаблоны (Templates), в свою очередь, описывают, как данные представить пользователю.

Разработанная система состоит из двух частей: серверная и клиентская. Реализованные клиентская часть — это интерфейс, с помощью которого пользователь может взаимодействовать с серверной частью.

Выбранные технологии для Frontend:

1. HTML5 - стандартный язык разметки для веб-страниц.

2. Bootstrap v5.1- свободный набор инструментов для создания сайтов и веб-приложений. Включает в себя HTML- и CSS-шаблоны оформления для типографики, веб-форм, кнопок, меток, блоков навигации и прочих компонентов веб-интерфейса, включая JavaScript-расширения.

3. JavaScript - это легковесный, интерпретируемый или JIT компилируемый, объектно-ориентированный язык с функциями первого класса. Наиболее широкое применение находит как язык сценариев веб-страниц.

Выбранные технологии для Backend:

1. Язык программирования Python 3.9.7.

2. Django 3.2.7 - свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

3. Для реализации машинного обучения был выбран фреймворк - scikit-learn, который является простым и эффективным инструментом для предиктивного анализа данных, основанный на NumPy, SciPy, и matplotlib.

4. Для разработки данного ПО была выбрана IDE – PyCharm 2021.2.2.

5. Для хранения данных была выбрана СУБД- SQLite 3.36.0.

На рисунке 1 представлена архитектура системы.

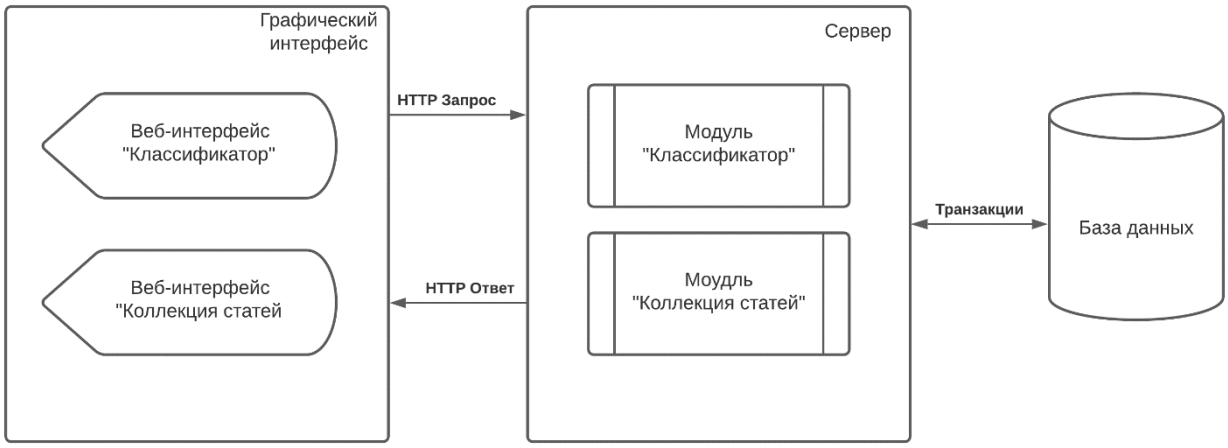

Рисунок 1 – Архитектура системы

4 ОПИСАНИЕ КЛАССИФИКАТОРА

4.1 Классы

В системе поддерживаются следующие классы:

1. спорт;
2. культура;
3. интернет и СМИ;
4. наука и техника;
5. экономика.

4.2 Описание алгоритма

Алгоритм работы классификатора состоит из 3 этапов:

1. Сбор данных.
2. Подготовка данных.
3. Машинной обучение.

Для сбора данных был произведен анализ новостных ресурсов с похожими категориями. При помощи парсинга и поиска был собран файл формата .csv, содержащий 2 поля: «наименование категории», «текст» и состоящий из 10000 статей. Разница в количестве статей по каждой категории приблизительно равна около 10%.

Перед классификацией текста требуется подготовить данные для тестовой и обучающий выборки. В системе подготовка данных происходит в 3 этапа:

1. Предобработка. Реализуется 3 методами:

- def remove_punctuation(text) – убирает пунктуацию;
- def remove_numbers(text) – убирает цифры;
- def remove_multiple_spaces(text) – убирает множественные пробелы.

Используемая библиотека: nltk.corpus, модуль: stopwords;

- def remove_stop_words(text) – убирает стоп – слова.

2. Стемминг текста. Метод:

- def stemming_text(text) – выделяет основу слова без его окончания. Используемая библиотека: nltk, модуль: SnowballStemmer.

3. Лемматизация текста. Метод:

- def lemmatize_text(text,mystem)- приводит глаголы в начальную форму. Используемая библиотека: pymystem3, модуль: Mystem.

Данные методы используются как для подготовки датасета нейронной сети, так и для текста, который нужно классифицировать.

После подготовки датасета, разделяем его на тестовую и обучающую выборку с помощью:

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Используемая библиотека: Scikit-learn, модуль: train_test_split

X = подготовленной текст; y = категория текста. Были выбраны значения для выборок:

- 70% - обучающая выборка;
- 30% - тестовая.

Для обучения модели используется библиотеку Scikit-learn.

В работе используется классификатор, который использует алгоритм метода опорных векторов. Основная идея метода заключается в построении гиперплоскости, разделяющей объекты выборки оптимальным способом. Алгоритм работает в предположении, что чем больше расстояние (зазор) между разделяющей гиперплоскостью и объектами разделяемых классов, тем меньше будет средняя ошибка классификатора.

Перед запуском обучения, реализуется конвейерная обработка, которая содержит 3 обработчика:

- CountVectorizer() – векторизует данные в матрицу;

- TfidfTransformer() – преобразует матрицу в нормализованное представление tf-idf;
- SGDClassifier() – классификатор SGD реализует регуляризованные линейные модели со стохастическим градиентным спуском.

Обучение реализуется с помощью метода: sgd.fit(X_train,y_train).

Затем предсказываем результат классификатора на тестовой выборке: y_pred = sgd.predict(X_test).

Точность работы классификатора представлена на рисунке 2.

```
*****Classifier Information******
The accuracy of the classifier: 0.947
                precision    recall  f1-score   support

Sport                0.94      0.88      0.90       610
Culture              0.96      0.97      0.96       591
Internet and media   0.96      0.96      0.96       584
Science and tech.    0.94      0.98      0.96       626
Economy              0.95      0.95      0.95       589

accuracy                               0.95      3000
macro avg            0.95      0.95      0.95      3000
weighted avg         0.95      0.95      0.95      3000
```

Рисунок 2 – Точность работы классификатора

Для того, чтобы убедиться, что данный алгоритм классификации является наилучшим, в процессе разработки дополнительно было разработано 2 классификатора:

1. Наивный байесовский классификатор.

Данный классификатор позволяет рассчитать апостериорную вероятность P(A | B) на основе P(A), P(B) и P(B | A).

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}$$

Где:

P(A | B) – апостериорная вероятность (что A из B истинно);

P(A) – априорная вероятность (независимая вероятность A);

P(B | A) – вероятность данного значения признака при данном классе. (что B из A истинно);

P(B) – априорная вероятность при значении нашего признака. (независимая вероятность B).

Запускаем обучение модели с помощью данного алгоритма, раскомментировав конвейер nb и строку nb.fit(X_train, y_train).

Результаты работы классификатора представлены на рисунке 3.

```
The accuracy of the classifier: 0.943
                      precision    recall  f1-score   support

          Sport            0.91      0.88      0.90       610
        Culture            0.97      0.95      0.96       591
Internet and media         0.96      0.94      0.95       584
Science and tech.          0.95      0.98      0.96       626
        Economy            0.93      0.97      0.95       589

       accuracy                               0.94      3000
      macro avg            0.94      0.94      0.94      3000
   weighted avg            0.94      0.94      0.94      3000
```

Рисунок 3 – Точность работы наивного байесовского классификатора

2. Классификатор, основанный на логистической регрессии.

Основная идея логистической регрессии заключается в том, что пространство исходных значений может быть разделено линейной границей (т.е. прямой) на две соответствующих классам области. В случае двух измерений — это просто прямая линия без изгибов. В случае трех — плоскость, и так далее. Эта граница задается в зависимости от имеющихся исходных данных и обучающего алгоритма. Чтобы все работало, точки исходных данных должны разделяться линейной границей на две вышеупомянутых области.

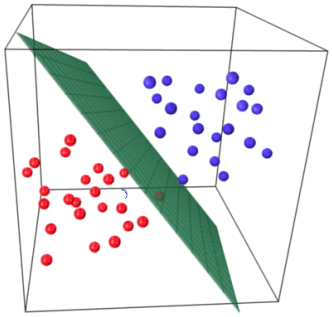

Рисунок 4 - Основная идея логистической регрессии

Запускаем обучение модели с помощью данного алгоритма, раскомментировав конвейер logreg и строку logreg.fit(X_train, y_train).

Результаты работы классификатора представлены на рисунке 4.

The accuracy of the classifier: 0.938

	precision	recall	f1-score	support
Sport	0.93	0.92	0.92	610
Culture	0.98	0.97	0.97	591
Internet and media	0.97	0.97	0.97	584
Science and tech.	0.96	0.98	0.97	626
Economy	0.96	0.96	0.96	589
accuracy			0.96	3000
macro avg	0.96	0.96	0.96	3000
weighted avg	0.96	0.96	0.96	3000

Рисунок 5 – Точность работы классификатора, основанного на алгоритме логистической регрессии

Исходя из полученных данных можно сделать вывод, что классификатор, использующий метод опорных векторов показывает лучшие результаты в классификации текста среди методов, представленных в данной пояснительной записке. Такой результат получен благодаря тщательной подготовки данных для классификатора и количеству этих же данных для каждой категории, используемой в классификаторе.

Разница в работе других классификаторов с классификатором, который используется в системе минимальная и составляет 0,1%.

Для того, чтобы добиться улучшений в показателях эффективности работы классификатора требуется пересмотреть данные для обучения. К примеру: по каждой категории определить одинаковый набор данных с приблизительно одинаковым количеством слов в тексте.

5 ОПИСАНИЕ ИНФРАСТРУКТУРЫ РАЗРАБОТКИ

В программном обеспечении используются методы, описанные ранее в данной пояснительной записке и методы, к которым закрыт доступ по соображениям безопасности исходных данных разработанного классификатора.

Для доступа к закрытым методам администратору или разработчику требуется открыть исполняемый файл Classifier.py, расскоментировать нужный вызов функции и запустить файл. Результат работы можно увидеть в консольном окне среды разработки или в терминале операционной системы, в зависимости от того, где запускается исполняемый файл.

В процессе разработки применялись следующие пакеты:

beautifulsoup4==4.10.0 - пакет Python для синтаксического анализа HTML и XML-документов.

Django==3.2.9 - свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

lxml==4.6.4 - многофункциональная и простая в использовании библиотека для обработки XML и HTML на языке Python

nltk==3.6.5 - пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python.

numpy==1.21.4 - это open-source модуль для python, который предоставляет общие математические и числовые операции в виде пре-скомпилированных, быстрых функций.

pandas==1.3.4 - программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня.

pymystem3==0.2.0 - предоставляет программный интерфейс к анализатору для языка программирования Python.

scikit-learn==1.0.1- библиотека машинного обучения свободного программного обеспечения для языка программирования Python.

Приложение собрано и опубликовано в репозитории системы контроля версий GitHub. [5]

Для скачивания приложения требуется ввести ссылку на репозиторий [5] в браузере, нажать кнопку «Code», затем нажать в выпадающем меню кнопку «Download ZIP». После вышеуказанных действий требуется разархивировать скаченные файлы в отдельную созданную папку и открыть созданную папку в IDE. [6]

Перед запуском приложения следует зайти в файл Settings.py и изменить настройки переменных:

1. Debug = False (True – по-умолчанию).

2. ALLOWED_HOSTS = ["127.0.0.1", "localhost"] (адреса по-умолчанию).

Для запуска сервера обязательно наличие интерпретатора Python версии 3.9.

Для запуска приложения требуется ввести несколько команд в терминале.

1. pip install requirements.txt – установит все необходимые библиотеки для работы ПО;

2. py manage.py runserver – запустит ПО по адресу 127.0.0.1.

6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

6.1 Метод тестирования

Тестирование данного программного обеспечения проводилось методом Black-box. Black-box тестирование – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. Метод тестирования «черного ящика» – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство. [4]

6.2 Сценарии тестирования

Результаты тестирования представлены в формате тест-кейсов в таблицах 1-12.

Таблица 1 – Тестирование верхнего меню интерфейса системы»

Предусловие		
Переход по ссылкам верхнего меню путём нажатия левой кнопкой мыши на соответствующую ссылку.		
Тестовый сценарий		
Действие	Ожидаемый результат	Фактический результат (выполнено, не выполнено, выполнено с ошибкой).
1. Нажать левой кнопкой мыши на ссылку “Главная”.	1. Переход на страницу “Главная”.	1. Выполнено.
2. Нажать левой кнопкой мыши на ссылку “Коллекция текстов”.	2. Переход на страницу “Коллекция текстов”.	2. Выполнено.
3. Нажать левой кнопкой мыши на ссылку “Добавить статью”.	3. Переход на страницу “Главная ”.	3. Выполнено.

Таблица 2 – Тестирование страницы «Главная »(URL: 127.0.0.1:8000)»

Предусловие		
Переход на адрес 127.0.0.1:8000. Классификатор.		
Тестовый сценарий		
Действие	Ожидаемый результат	Фактический результат (выполнено, не выполнено, выполнено с ошибкой).
1. Ввести текст в соответствующее поле и нажать левой кнопкой мыши на кнопку классифицировать.	1. Вывод в соответствующем поле категорию, к которой относится введённый текст.	1. Выполнено.
2. Нажать левой кнопкой мыши на ссылку под названием “Тык”.	2. Переход на страницу с информацией о классификаторе.	2. Выполнено.

Таблица 3 – Тестирование страницы «Коллекция текстов»(URL: 127.0.0.1:8000/articles)»

Предусловие		
Переход на адрес 127.0.0.1:8000/articles. Коллекция текстов		
Тестовый сценарий		
Действие	Ожидаемый результат	Фактический результат (выполнено, не выполнено, выполнено с ошибкой).
1. Ввести текст в соответствующее поле и нажать левой кнопкой мыши на кнопку классифицировать.	1. Вывод в соответствующем поле категорию, к которой относится введённый текст.	1. Выполнено.
2. Нажать левой кнопкой мыши на ссылку под названием “Тык”.	2. Переход на страницу с информацией о классификаторе.	2. Выполнено.

Таблица 4 – Тестирование страницы «Добавить статью»(URL: 127.0.0.1:8000/addarticle)»

Предусловие		
Переход на адрес 127.0.0.1:8000/addarticle. Добавить статью		
Тестовый сценарий		
Действие	Ожидаемый результат	Фактический результат (выполнено, не выполнено, выполнено с ошибкой).
1. Заполнить все обязательные поля и нажать левой кнопкой мыши на кнопку “Добавить”.	1. Переход на страницу с подтверждением результата.	1. Выполнено.
2. Заполнить все обязательные поля в некорректном формате и нажать левой кнопкой мыши на кнопку “Добавить”.	2. Вывод ошибки с пояснением о некорректном вводе данных в соответствующем поле, где данные заведомо ввели некорректно.	2. Выполнено.
3. Клик левой кнопкой мыши на кнопку “Назад”.	3. Переход на страницу, предшествующей этой исходя из истории пользователя в системе.	3. Не выполнено.

Таблица 5 – Тестирование страницы «Коллекция текстов»(URL: 127.0.0.1:8000/article/#)» # - номер статьи

Предусловие		
Переход на адрес 127.0.0.1:8000/ article/#. Определенная статья.		
Тестовый сценарий		
Действие	Ожидаемый результат	Фактический результат (выполнено, не выполнено, выполнено с ошибкой).
1. Нажать левой кнопкой мыши на кнопку “Удалить”.	1. Отображение кнопки с подтверждением.	1. Выполнено.
2. Нажать левой кнопкой мыши на кнопку “Подтвердить удаление”.	2. Переход на страницу с информационным сообщением об удалении статьи.	2. Выполнено.
3. Нажать левой кнопкой мыши на кнопку “Обновить”.	3. Отображение кнопки с подтверждением.	3. Выполнено.
4. Нажать левой кнопкой мыши на кнопку “Подтвердить обновление”.	4. Вывод сообщения об успешном обновлении статьи.	4. Не выполнено.
5. Клик левой кнопкой мыши на кнопку “Назад”.	5. Переход на страницу, предшествующей этой исходя из истории пользователя в системе.	5. Выполнено.
5. Клик левой кнопкой мыши на кнопку “Следующая страница”.	5. Переход на следующую страницу с коллекцией текстов.	5. Не выполнено.

Таблица 6 – Тестирование функционала фильтрации текста

Предусловие		
Переход на адрес 127.0.0.1:8000/articles и применение фильтра для поиска нужных статей.		
Тестовый сценарий		
Действие	Ожидаемый результат	Фактический результат (выполнено, не выполнено, выполнено с ошибкой).
1. Нажать левой кнопкой мыши на кнопку “Найти”.	1. Отображение всех новостных статей.	1. Выполнено.
2. Выбрать категорию и нажать левой кнопкой мыши на кнопку “Найти”.	2. Отображение статей с выбранной категорией.	2. Выполнено.
3. Выбрать диапазон дат публикации и нажать левой кнопкой мыши на кнопку “Найти”.	3. Отображение статей в выбранном диапазоне.	3. Выполнено.
4. Ввести некорректно даты и нажать левой кнопкой мыши на кнопку “Найти”.	4. Отображение ошибки о некорректной дате.	4. Выполнено.
5. Выбрать сортировку, её направление и нажать левой кнопкой мыши на кнопку “Найти”.	5. Отображение статей в выбранном направлении сортировки.	5. Выполнено.
6. Выбрать сортировку, не выбрав направление и нажать левой кнопкой мыши на кнопку “Найти”.	6. Отображение ошибки о том, что направление сортировка не было выбрано.	6. Выполнено.

6.2 Результаты тестирования

В результате тестирования методом черного ящика, состоящего из 22 сценариев успешно были выполнены 19 сценариев. 3 сценария не прошли проверку, в результате чего алгоритмы, которые используется в данных сценариях были пересмотрены и изменены.

7 ИНТЕРФЕЙС И ВОЗМОЖНОСТИ СИСТЕМЫ

Веб-приложение можно разделить на две подсистемы, которые могут работать как совместно, так и отдельно друг от друга.

При старте сервера, для пользователя главной страницей при запуске/открытия веб-приложения является «Классификатор текста» (рис.3). На данной странице пользователь может в специальном поле ввести текст, который нужно классифицировать и, нажав кнопку «Классифицировать» левой кнопкой мыши получить ответ в виде категории (или класса), к которой принадлежит текст (рис.4).

В шапке данной страницы отображается меню системы, посредством нажатия левой кнопки мыши, пользователь может перемещаться по страницам приложения. Ниже верхнего меню отображается уведомление о существующих категориях/классах, которые может определить классификатор для вновь веденного текста.

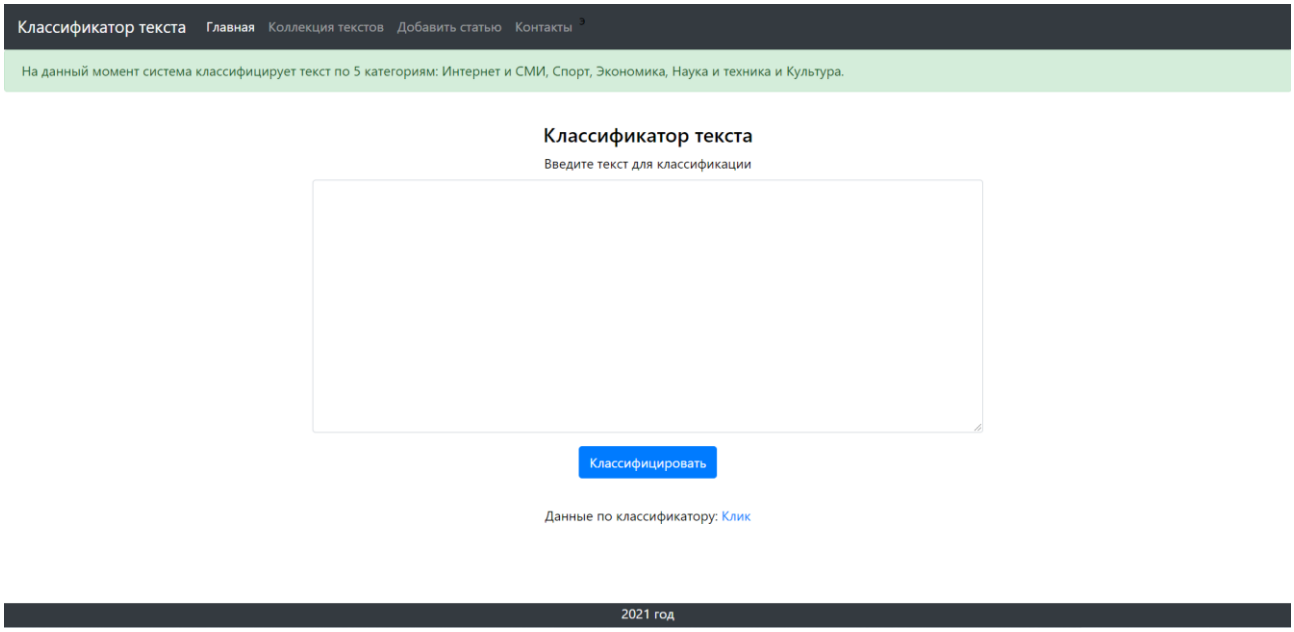

Рисунок 3 – Веб-интерфейс страницы «Классификатор текста»

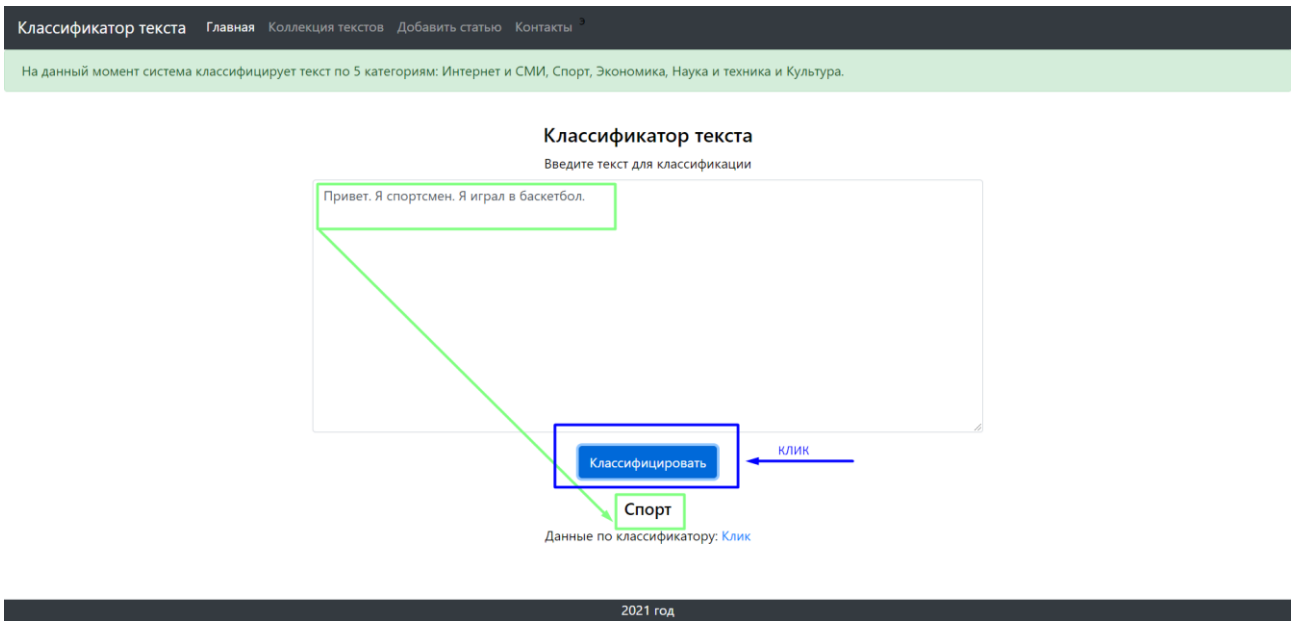

Рисунок 4 – Веб-интерфейс страницы «Классификатор текста». Определение принадлежности текста к классу/категории

Перемещаясь по верхнему меню, для пользователя предусмотрена возможность открыть страницу с коллекцией текстов (рис. 5). Здесь можно найти интересующую новостную статью путём применения фильтра. Ниже меню отображено уведомление о том, как использовать фильтр. На рисунке 6 отображен пример применения фильтра, где пользователю необходимо найти статьи с категорией «Спорт» и отображение данных статей с сортировкой по возрастанию.

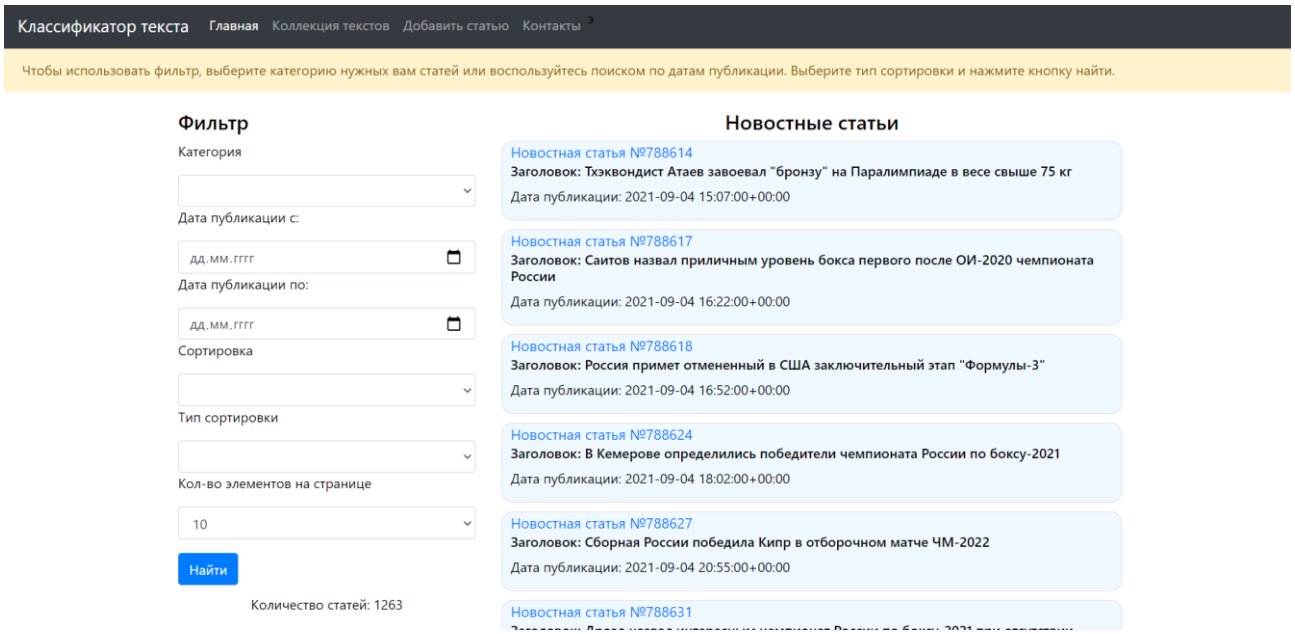

Рисунок 5 – Веб-интерфейс страницы «Коллекция текстов».

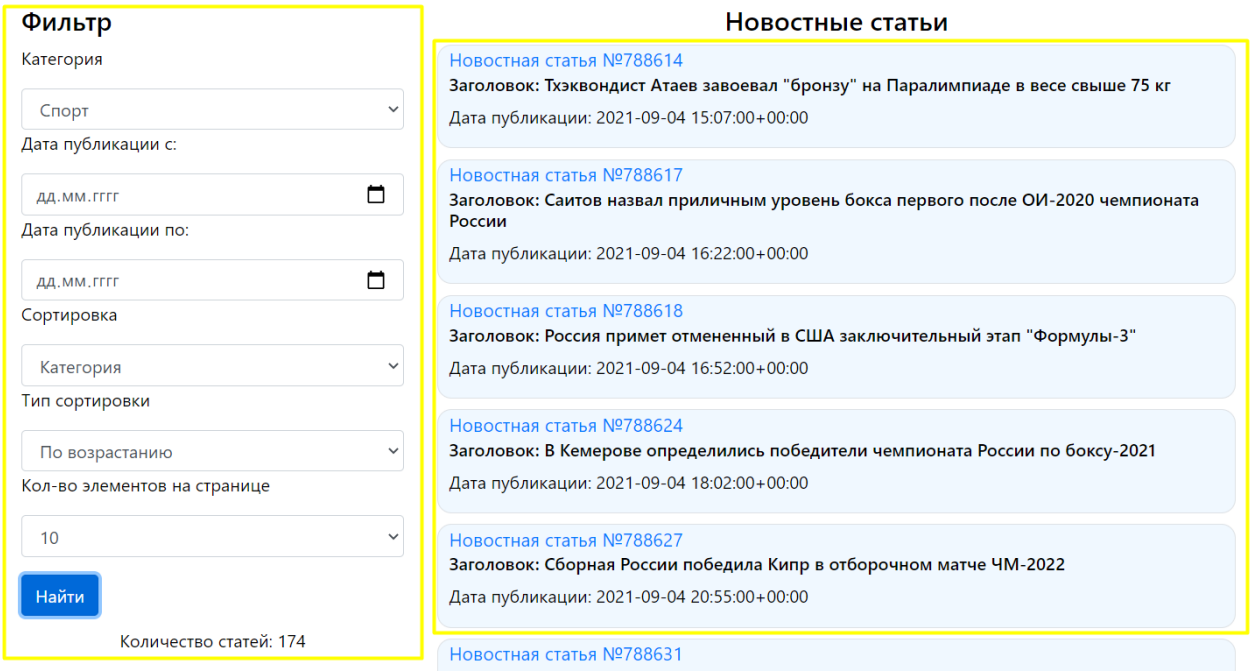

Рисунок 6 – Применения фильтра на странице «Коллекция текстов».

Пользователь может просмотреть информацию по статье в подробном содержании путём клика левой кнопкой мыши на ссылку новостной статьи. Отобразится страница статьи, где пользователю предоставляется возможность изменить, удалить статью или перейти назад (рис. 7).

Для того, чтобы удалить или обновить (изменить содержание) новостную статью требуется подтвердить указанные действия путём нажатия левой кнопкой мыши на соответствующую кнопку (рис. 8).

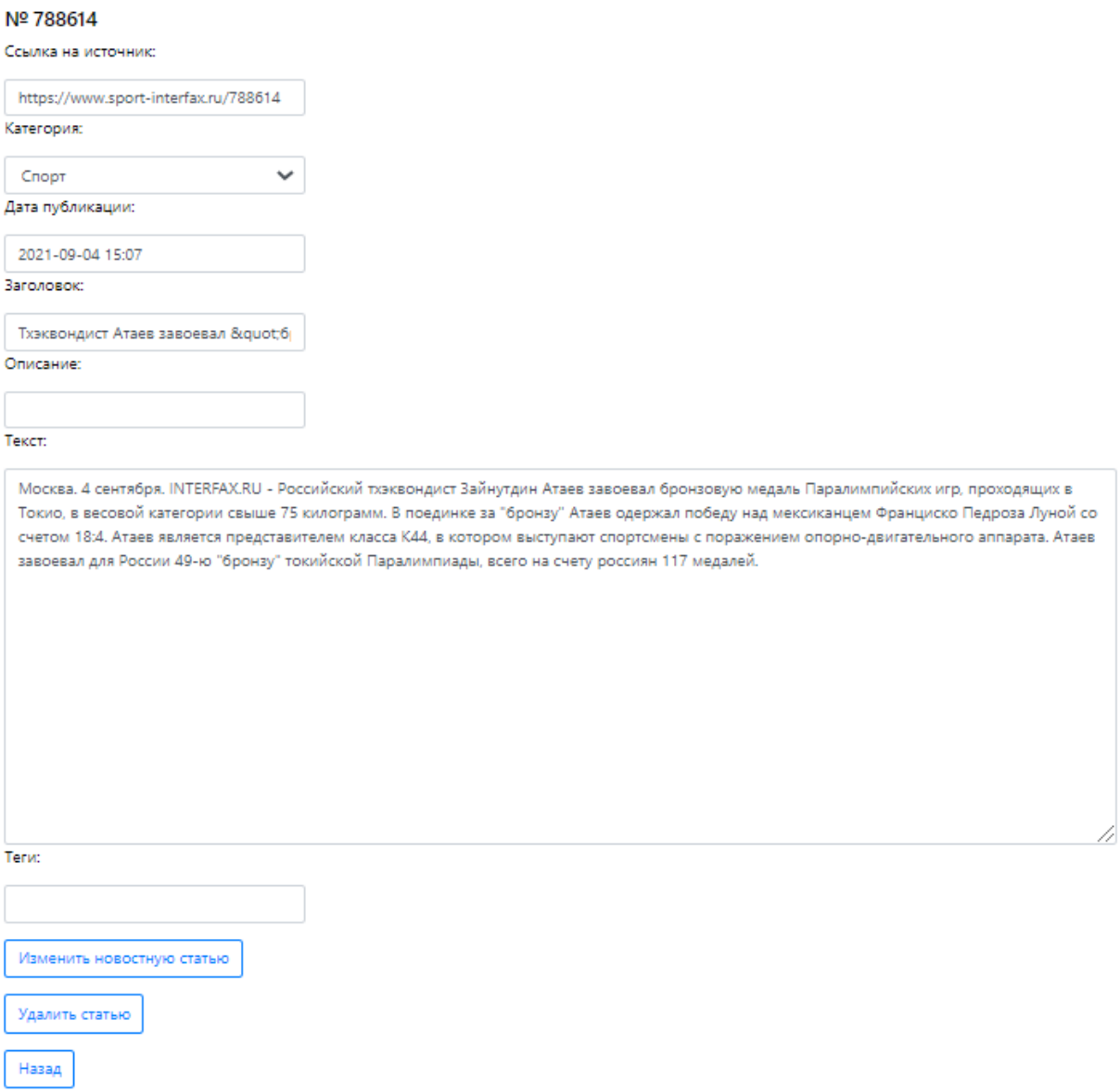

Рисунок 7 – Веб-интерфейс страницы «Новостная статья»

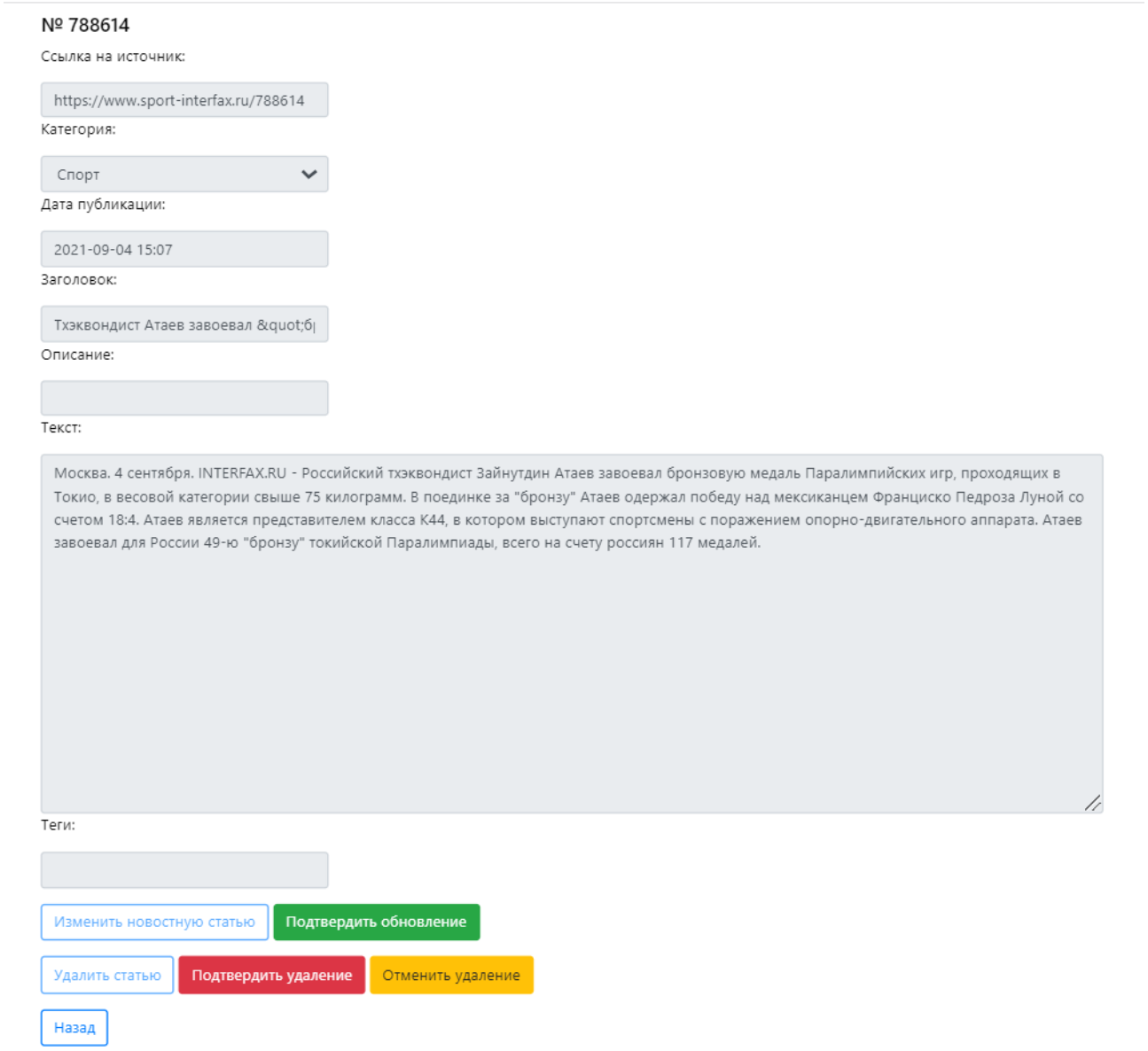

Рисунок 8 – Подтверждение об удалении/обновлении новостной статьи

Пользователем может быть открыта страница «Добавить статью» (рис.9), где при заполнении в корректном формате всех полей формы и, нажав кнопку «Добавить статью» левой кнопкой мыши, в базе появится новая новостная статья, добавленная пользователем системы. При заполнении полей в некорректном формате, системы попытается исключить неправильную форму ввода и подскажет пользователю, где он совершил ошибку.

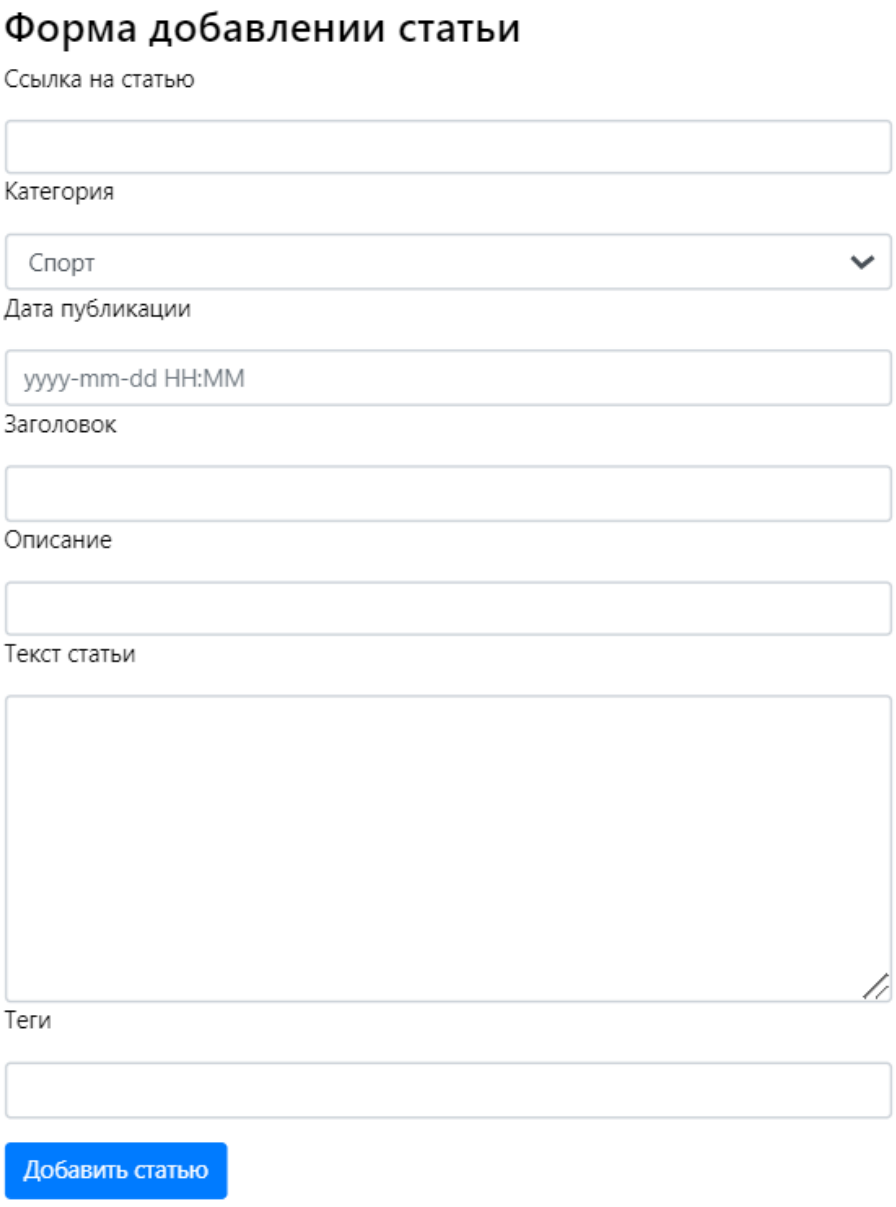

Рисунок 9 – Веб-интерфейс страницы «Добавить статью»

8 АНАЛИЗ РЕЗУЛЬТАТОВ

В результате проделанной работы было разработано клиент-серверное веб-приложение, с помощью которого можно взаимодействовать с коллекцией текстов путём редактирования, добавления, удаления текстов и классифицировать текст по 5 категориям.

Хорошо подготовленный датасет и его обработка позволила добиться точности работы классификатора в ~95%, что является отличным результатом.

Взаимодействие с коллекцией текстов и с классификатором происходит в интуитивно-понятном интерфейсе для пользователя, что является немало важным критериям для оценки данной системы.

Система попытается обработать все ошибки пользователя, когда пользователь будет вносить изменения в соответствующие поля. Путём пользовательского тестирования (betta test), в программное обеспечение будет добавляться новый функционал по обработке ошибок пользователя.

Система масштабируемая, то есть, исходя из требований заказчика, в систему может быть добавлен новый функционал или переработан старый, не затрагивая все узлы системы.

Исходя из результатов тестирования, можно сделать вывод о том, что система в работоспособном состоянии.

При написании данного курсового проекта был получен опыт в реализации классификатора путём применения алгоритмов машинного обучения, фронтенда и бэкенда, разработанного на фреймворке Django+Python.

Заявленные требования к курсовой работе выполнены полностью в соответствии с требованиями заказчика. Работа предоставлена в срок.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Django. Разработка веб-приложений на Python | Форсье Джефф [Электронный ресурс] // zhurnalonlain.ru: информационно – справочный портал. – Режим доступа: https://zhurnalonlain.ru/programming/5929-django-razrabotka-veb-prilozhenii-na-python-2017-uesli-chan.html (Дата обращения: 14.12.2021)

2. Django. Адриан Головатый, Джейкоб Каплан-Мосс [Электронный ресурс] // step-develop.com: информационно – справочный портал. – Режим доступа: https://www.step-develop.com/downloads/django-book-02.pdf (Дата обращения: 14.12.2021)

3. «Machine Learning, Neural and Statistical Classification» D. Michie, D.J. Spiegelhalter, C.C. Taylor. [Электронный ресурс] // researchgate.net: электронная библиотека. – Режим доступа: https://www.researchgate.net/publication/2335004_Machine_Learning_Neural_and_Statistical_Classification (Дата обращения: 14.12.2021)

4. Особенности тестирования «черного ящика» [Электронный ресурс] // quality-lab.ru: информационно-справочный портал. – Режим доступа: https://quality-lab.ru/blog/key-principles-of-black-box-testing/ (Дата обращения: 14.12.2021)

5. Проект системы TextClassifierWebService [Электронный ресурс] // github.com: сервис онлайн хостинга репозиториев. – Режим доступа: https://github.com/Mitsich/TextClassifierWebServiceRelease (Дата обращения: 14.12.2021)

6. Выбираем самый удобный редактор кода Python [Электронный ресурс] //habr.com: Интернет-ресурс для IT-специалистов. – Режим доступа: https://habr.com/ru/company/skillfactory/blog/521838/ (Дата обращения: 14.12.2021)