МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Кафедра 319 «Системы интеллектуального мониторинга»

КУРСОВАЯ РАБОТА

по дисциплине «Технология разработки программного обеспечения»

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

Студент _____ Пожидаев Е.В.

Группа М3О – 221М – 19

Руководитель _____ Полицына Е.В.

Оценка _____ Дата защиты «____» декабря 2020 г.

Москва 2020

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Кафедра 319 «Системы интеллектуального мониторинга»

З А Д А Н И Е

на курсовую работу по дисциплине
Технология разработки программного обеспечения

Студент М3О – 221М – 19 Пожидаев Егор Витальевич
(№ группы, Ф. И. О.)

Тема Проектирование и разработка веб-приложения классификации новостей с применением методов машинного обучения

Перечень вопросов, подлежащих разработке в курсовой работе:

1. Проектирование архитектуры системы и выбор средств разработки
2. Проектирование и реализация ядра системы – классификатора
3. Проектирование и реализация фронтенда
4. Проектирование и реализация бекенда

Рекомендуемая литература

1. Django [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/. – Заглавие с экрана. – (Дата обращения 25.12.20).
2. Scikit-learn [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/. – Заглавие с экрана. – (Дата обращения 25.12.20).

Задание выдано « 12 » сентября 2020 г.
Руководитель Полицына Екатерина Валерьевна, к.т.н., доцент кафедры 319 МАИ
(Ф. И. О., должность, подпись)
Студент Пожидаев Е.В.
(подпись)

СОДЕРЖАНИЕ

1 ОПИСАНИЕ ВОЗМОЖНОСТЕЙ ПРИЛОЖЕНИЯ

1.1 Общие сведения

Веб-приложение классификации новостных статей представляет веб-сервис, который позволяет классифицировать новости и управлять коллекцией собранных новостей с информационного портала «interfax.ru».

1.2 Назначение и цель

С помощью веб-приложения возможно определить, к какой категории относится новостная статья, введенная пользователем. Сервис предоставляет возможность просмотреть собранные новостные статьи, которые представляли собой входные данные для обучения классификатора с возможностью их фильтрации. Также веб-приложение позволяет удалять, изменять собранные новостные статьи. У веб-приложения есть возможность добавить новостную статью в коллекцию.

2 ТРЕБОВАНИЯ К ПРИЛОЖЕНИЮ

2.1 Функциональные требования

Функциональные требования к системе:

- добавление новостной статьи;
- удаление новостной статьи;
- изменение новостной статьи;
- вывод новостных статей по страницам;
- фильтрация новостных статей по категориям и диапазону даты публикации;
- классификация текста по категории новостной статьи с помощью методов машинного обучения;
- обновление модели классификатора.

2.2 Нефункциональные требования

Система должна быть реализована на базе клиент-серверной архитектуры и представлять из себя веб-сервис для классификации новостных статей с графическим пользовательским интерфейсом.

Веб-приложение должно быть разработано с помощью следующих технических средств:

- backend должен быть реализован на языке программирования Python с помощью фреймворка Django;
- СУБД SQlite;
- frontend должен быть реализован с помощью HTML и языка программирования Javascript;
- проект и вся необходимая документация и данные по нему должны храниться в системе контроля версий git на сайте сервиса GitHub.

Система должна быть отказоустойчивой при работе пользователя с системой.

3 АРХИТЕКТУРА СИСТЕМЫ

3.1 Схема архитектуры

Архитектура разработанного приложения изображена на рисунке 1.

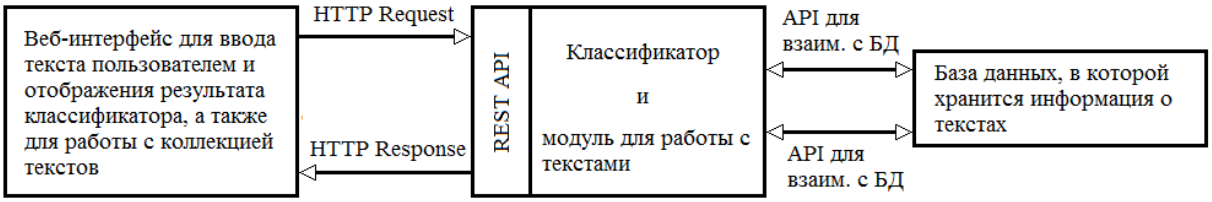

Рисунок 1 – Архитектура приложения

Frontend часть приложения, представляющая собой веб-интерфейс для ввода текста новости и работы с коллекцией текстов взаимодействует с backend частью приложения посредством HTTP запросов и ответов. Backend часть приложения состоит из базы данных, в которой хранится информация о новостных статьях, классификатора и модуля для работы с коллекцией новостей. Классификатор и модуль для работы с коллекцией новостей взаимодействует с базой данных с помощью API для взаимодействия с БД.

3.2 Протоколы взаимодействия

Разработанная система является RESTful веб-сервисом. Используемые в системе API:

- Страница классификатора.

GET /

Возвращает html-документ страницы классификатора.

- Страница для работы с коллекцией текстов.

GET /collection

Возвращает html-документ страницы для работы с коллекцией текстов.

- Страница новостной статьи.

GET /article/{articleId}

Возвращает html-документ страницы новостной статьи.

- Фильтрация новостных статей.

POST /api/articles/filter

Данная точка входа фильтрует новостные статьи по категориям и по диапазону даты публикации, при этом позволяет отсортировать полученную выборку по данным полям, а также провести выборку по страницам, с возможностью задания количества элементов на них. Возвращает в теле ответа json, который содержит информацию о новостных статьях, а также их количестве. Json-схема json ответа фильтрации новостных статей представлена в листинге 1. Точка входа получает в теле запроса json, который содержит информацию о параметрах для фильтрации. Json-схема json запроса представлена в листинге 2.

Листинг 1 – Json-схема при ответе точки входа для фильтрации новостных статей.

```
{
    “$schema”: “http://json-schema.org/draft-04/schema#”,
    “title”: “filterResult”,
    “description”: “information about filtered articles”,

    “properties”:{
        “len”:{
            “type”: “integer”
        },
        “articles”:{
            “type”: “array”,
            “items”:{
                “type”: “object”,
                “properties”:{
                    “id”:{
                        “type”: “integer”
                    },
                    “title”:{
                        “type”: “string”
                    },
                    “category”:{
                        “type”: “string”
                    },
                    “datetime”:{
                        “type”: “string”
                    },
                    “description”:{
                        “type”: “string”
```

```
                    },
                    “tags”:{
                        “type”: “array”,
                        “items”:{
                            “type”: “string”
                        }
                    }
                }
            }
        }
    }
}
```

Листинг 2 – Json-схема при запросе точки входа для фильтрации новостных статей.

```
{
    “$schema”: “http://json-schema.org/draft-04/schema#”,
    “title”: “filterParameters”,
    “description”: “information about filter for articles”,

    “properties”:{
        “category”:{
            “type”: [“string”, “null”]
        },
        “dateStart”:{
            “type”: [“string”, “null”]
        },
        “dateEnd”:{
            “type”: [“string”, “null”]
        },
        “sort”:{
            “type”: [“string”, “null”]
        },
        “sortType”:{
            “type”: [“string”, “null”]
        },
        “page”:{
            “type”: [“string”, “null”]
        },
        “pageLimit”:{
            “type”: [“string”, “null”]
        }
    }
}
```

- Добавление новостной статьи.

POST /api/articles

Возвращает html ответ со статусом 200 при успешном добавлении новостной статьи. Точка входа получает в теле запроса json, который содержит

информацию о добавляемой статье. Json-схема json запроса представлена в листинге 3.

Листинг 3 – Json-схема при запросе точки входа для добавления новостной статьи.

```
{
     “$schema”: “http://json-schema.org/draft-04/schema#”,
     “title”: “articleInformationForAdd”,
     “description”: “information about article for add to collection”,

     “properties”:{
          “source”:{
               “type”: “string”
          },
          “category”:{
               “type”: “string”
          },
          “datetime”:{
               “type”: “string”
          },
          “title”:{
               “type”: “string”
          },
          “description”:{
               “type”: [“string”, “null”]
          },
          “text”:{
               “type”: “string”
          },
          “tags”:{
               “type”: [“array”, “null”],
               “items”:{
                    “type”: “string”
               }
          }
     }
}
```

- Вывод новостной статьи.

GET /api/articles/{articleId}

Возвращает в теле ответа json, который содержит информацию о новостной статье. Json-схема json овета представлена в листинге 4.

Листинг 4 – Json-схема при ответе точки входа для вывода новостной статьи.

```
{
     “$schema”: “http://json-schema.org/draft-04/schema#”,
```

```
    "title": "articleInformationForGet",
    "description": "information about output article",

    "properties":{
        "source":{
            "type": "string"
        },
        "category":{
            "type": "string"
        },
        "datetime":{
            "type": "string"
        },
        "title":{
            "type": "string"
        },
        "description":{
            "type": "string"
        },
        "text":{
            "type": "string"
        },
        "tags":{
            "type": "array"
            "items":{
                "type": "string"
            }
        }
    }
}
```

- Изменение новостной статьи.

PUT /api/articles/{articleId}

Возвращает html ответ со статусом 200 при успешном изменении новостной статьи. Точка входа в теле запроса получает json, который содержит информацию об измененной новостной статье. Json-схема json запроса представлена в листинге 5.

Листинг 5 – Json-схема при ответе точки входа для изменения новостной статьи.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "articleInformationForUpdate",
    "description": "information about article for update",

    "properties":{
        "source":{
            "type": "string"
```

```
            },
            “category”:{
                “type”: “string”
            },
            “datetime”:{
                “type”: “string”
            },
            “title”:{
                “type”: “string”
            },
            “description”:{
                “type”: [“string”, “null”]
            },
            “text”:{
                “type”: “string”
            },
            “tags”:{
                “type”: [“array”, “null”],
                “items”:{
                    “type”: “string”
                }
            }
        }
}
```

- Удаление новостной статьи.

DELETE /api/articles/{articleId}

Возвращает html ответ со статусом 200 при успешном удалении новостной статьи.

- Получение информации о классификаторе.

GET /api/classifier

Возвращает в теле ответа json, который содержит информацию о классификаторе. Json-схема json ответа представлена в листинге 6.

Листинг 6 – Json-схема при ответе точки входа для получения информации о классификаторе.

```
{
    “$schema”: “http://json-schema.org/draft-04/schema#”,
    “title”: “informationClassifier”,
    “description”: “information about accuracy of classifier”,

    “properties”:{
        “precision”:{
            “type”: “array”,
            “items”:{
```

```
                    “type”: “number”
               }
          },
          “precisionMacro”:{
               “type”: “number”
          },
          “precisionMicro”:{
               “type”: “number”
          },
          “recall”:{
               “type”: “array”,
               “items”:{
                    “type”: “number”
               }
          },
          “recallMacro”:{
               “type”: “number”
          },
          “recallMicro”:{
               “type”: “number”
          },
          “f1Score”:{
               “type”: “array”,
               “items”:{
                    “type”: “number”
               }
          },
          “f1ScoreMacro”:{
               “type”: “number”
          },
          “f1ScoreMicro”:{
               “type”: “number”
          },
          “accuracy”:{
               “type”: “number”
          },
          “categoryList”:{
               “type”: “array”,
               “items”:{
                    “type”: “string”
               }
          }
     }
}
```

- Классификация новостной статьи.

POST /api/classifier

Возвращает в теле ответа json, который содержит категорию – результат работы классификатора. Данная точка входа в теле запроса получает json, который содержит текст классифицируемой новостной статьи.

Json-схема json ответа представлена в листинге 7. Json-схема json запроса представлена в листинге 8.

Листинг 7 – Json-схема при ответе точки входа для классификации новостной статьи.

```
{
     “$schema”: “http://json-schema.org/draft-04/schema#”,
     “title”: “classifierResult”,
     “description”: “Result of classifier”,

     “properties”:{
          “category”:{
               “type”: “string”
          }
     }
}
```

Листинг 8 – Json-схема при запросе токи входа для классификации новостной статьи.

```
{
     “$schema”: “http://json-schema.org/draft-04/schema#”,
     “title”: “TextForclassifier”,

     “properties”:{
          “text”:{
               “type”: “string”
          }
     }
}
```

- Обновление модели классификатора.

PUT /api/classifier/fit

Возвращает html ответ со статусом 200 при успешном обновлении модели классификатора.

Для взаимодействия с базой данных используется ORM, которая является частью фреймворка Django.

3.3 Используемые технологии

Для реализации серверной части использовался язык программирования Python, а также фреймворк Django.

В качестве СУБД используется SQlite.

Для создания модели классификатора используется модуль языка Python scikit-learn.

В качестве средства реализации пользовательского интерфейса используется HTML и Javascript.

В качестве системы контроля версий использовался сервис GitHub, представляющий из себя графический интерфейс для технологии git – распределенной системы управления версиями.

3.4 Паттерны проектирования

При разработке использовался паттерн MVC, который позволяет разрабатывать бизнес-логику и визуальное представление отдельно.

4 ОПИСАНИЕ КЛАССИФИКАТОРА

4.1 Классы

Классификатор определяет следующие классы:

- в мире;
- в России;
- Москва;
- культура;
- спорт;
- экономика.

4.2 Вектор признаков

В качестве модели представления текста новостных статей использовался метод bag-of-words. При этом проводилась начальная подготовка текста: удалялись лишние неинформативные символы, а также различные стоп-слова. Вектором признаков является масштабированное отношение количества вхождений слова среди всех новостей на количество слов в тексте по методу tf-idf.

4.3 Модель машинного обучения классификатора

В качестве модели машинного обучения был выбран метод опорных векторов.

4.4 Обучающая и тестовая коллекция документов

В качестве обучающей коллекции были взяты 700 новостных статей по каждой из категории из общей коллекции текстов. Таким образом, классификатор обучался на 4200 новостных статьях.

В качестве тестовой коллекции были взяты 300 новостных статей по каждой из категории из общей коллекции текстов. Таким образом классификатор тестировался на 1800 новостных статьях.

4.5 Оценка точности классификации

Оценка точности разработанного классификатора оценивалась по нескольким критериям: точности (precision), полноте (recall), F-мере и точности (accuracy).

Точность (accuracy) классификации равна 83%. Оценки точности по категориям представлены в таблице 1.

Таблица 1 – Оценка точности обученного классификатора по категориям

Категория	Точность (precision)	Полнота (recall)	F-мера
В мире	0.79	0.8	0.8
В России	0.79	0.50	0.61
Москва	0.83	0.83	0.83
Культура	0.86	0.94	0.9
Спорт	0.96	0.97	0.97
Экономика	0.75	0.92	0.82
Взвешенное среднее	0.83	0.83	0.83

5 ОПИСАНИЕ ИНФРАСТРУКТУРЫ РАЗРАБОТКИ

5.1 Система контроля версий

Проектирование и реализация приложения велась локально на компьютере разработчика при помощи Microsoft Visual Studio 2017. При разработке системы использовалась система контроля версий git. Исходный код хранится локально на компьютере разработчика, и удаленно, на сайте сервиса GitHub: https://github.com/XaZdarova/Classifier.

5.2 Сборка и развертывание приложения

Для успешного запуска веб-приложения необходимо наличие интерпретатора Python 3.0 (64 бит) или выше, а также операционной системы Windows 7 и выше. Необходимо скачать архив с проектом, разархивировать папку ClassifierProject. Далее через командную строку активировать виртуальное окружение python с помощь команд:

- cd {pathTo}\ClassifierProject;
- env\Scripts\activate.bat.

После активации окружения, достаточно запустить web-приложение по данной команде: manage.py runserver.

5.3 Обновление модели классификатора

Для обновления модели классификатора, достаточно послать POST запрос по ссылке /api/classifier/fit.

6 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Для тестирования системы применялось ручное тестирование. Тестирование проводилось методом черного ящика.

В процессе тестирования использовался веб-браузер Google Chrome версии. Основные тестовые сценарии и результаты по ним представлены в таблице 2.

Таблица 2 – Основные тестовые сценарии и результаты тестирования

Сценарий тестирования	Результат тестирования
Ввод текста новостной статьи для классификации и последующая его классификация	Был выведен корректный результат классификации текста
Ввод пустого текста новостной статьи для классификации и последующая его классификация	Было выведено сообщение о том, что был введен пустой текст
Добавление новой новостной статьи	Было выведено сообщение о том, что статья успешно добавлена.
Добавление новой новостной статьи с некорректными параметрами	Было выведено сообщение о том, что параметры некорректны
Изменение новостной статьи	Было выведено сообщение о том, что статья успешно изменена
Изменение новостной статьи с некорректными параметрами	Было выведено сообщение о том, что параметры некорректны
Удаление новостной статьи	Было выведено сообщение о том, что статья удалена
Вывод страницы с новостными статьями	При загрузке страницы была выведена первая страница с коллекцией новостей
Вывод новостной статьи	При загрузке страницы с новостной статье, информация по ней была успешно отображена
Вывод несуществующей новостной статьи	Было выведено сообщение о том, что такой статьи не существует

7 ИНТЕРФЕЙС И ВОЗМОЖНОСТИ СИСТЕМЫ

Система предоставляет пользователю классифицировать текст и просматривать коллекцию новостных статей, а также производить операции с отдельными новостями.

Классификация текста происходит при нажатии кнопки «Классифицировать». Интерфейс страницы классификации текстов представлен на рисунке 2.

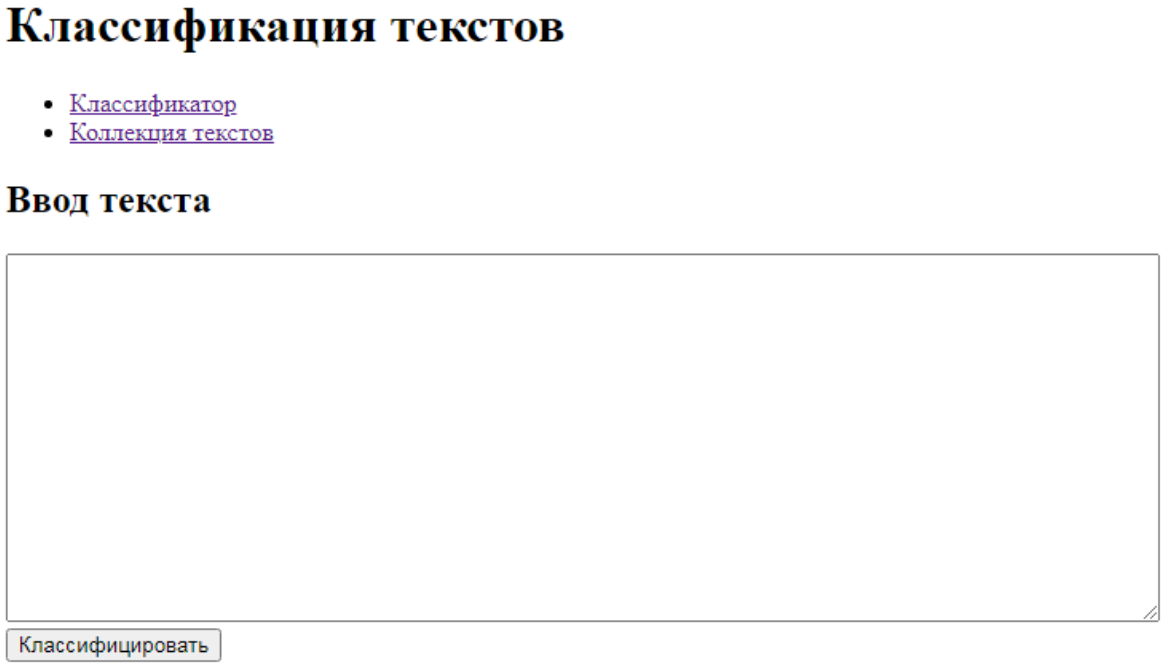

Рисунок 2 – Интерфейс классификации новостной статьи

Для фильтрации коллекции текстов необходимо задать желаемые параметры и нажать кнопку «Найти». Интерфейс для работы с коллекцией новостных статей представлен на рисунке 3.

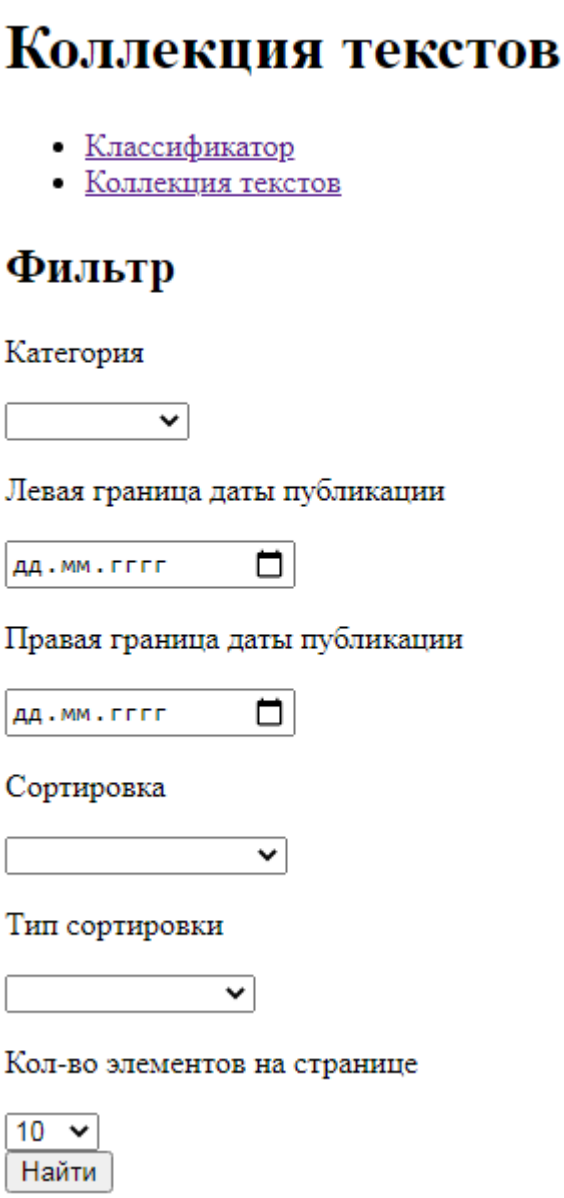

Рисунок 3 – Интерфейс для работы с коллекцией новостей

Найденные новостные статьи можно просматривать постранично, переключаясь между страницами с помощью кнопок «Предыдущая» и «Следующая», а также переходить по предоставленным ссылкам для получения более детальной информации, изменения или удаления статьи. Для изменения используется кнопка «Изменить новостную статью», а для удаления «Удалить новостную статью». Интерфейс для отображения коллекции новостей представлен на рисунке 4. Интерфейс для работы с отдельной новостью представлен на рисунке 5.

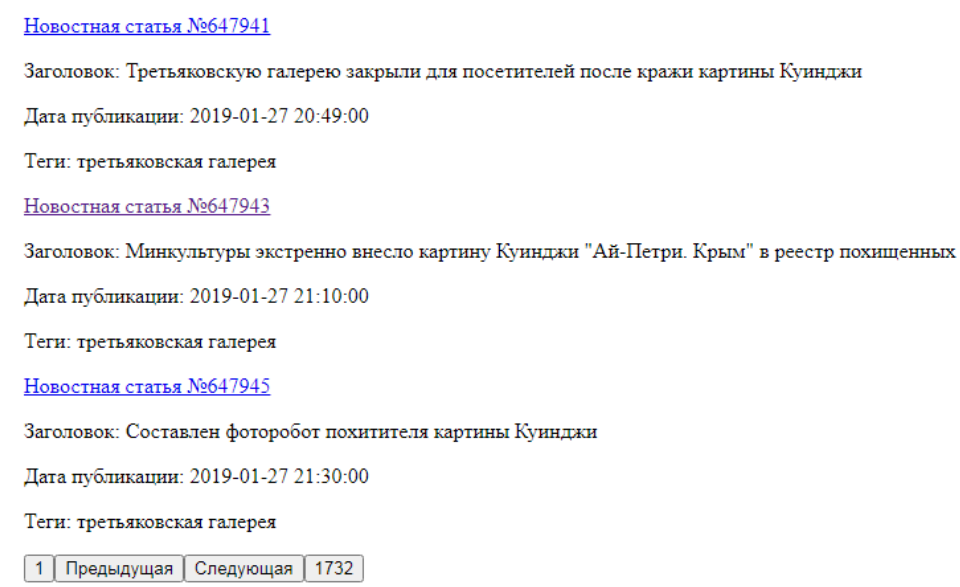

Рисунок 4 – Интерфейс для отображения коллекции новостей

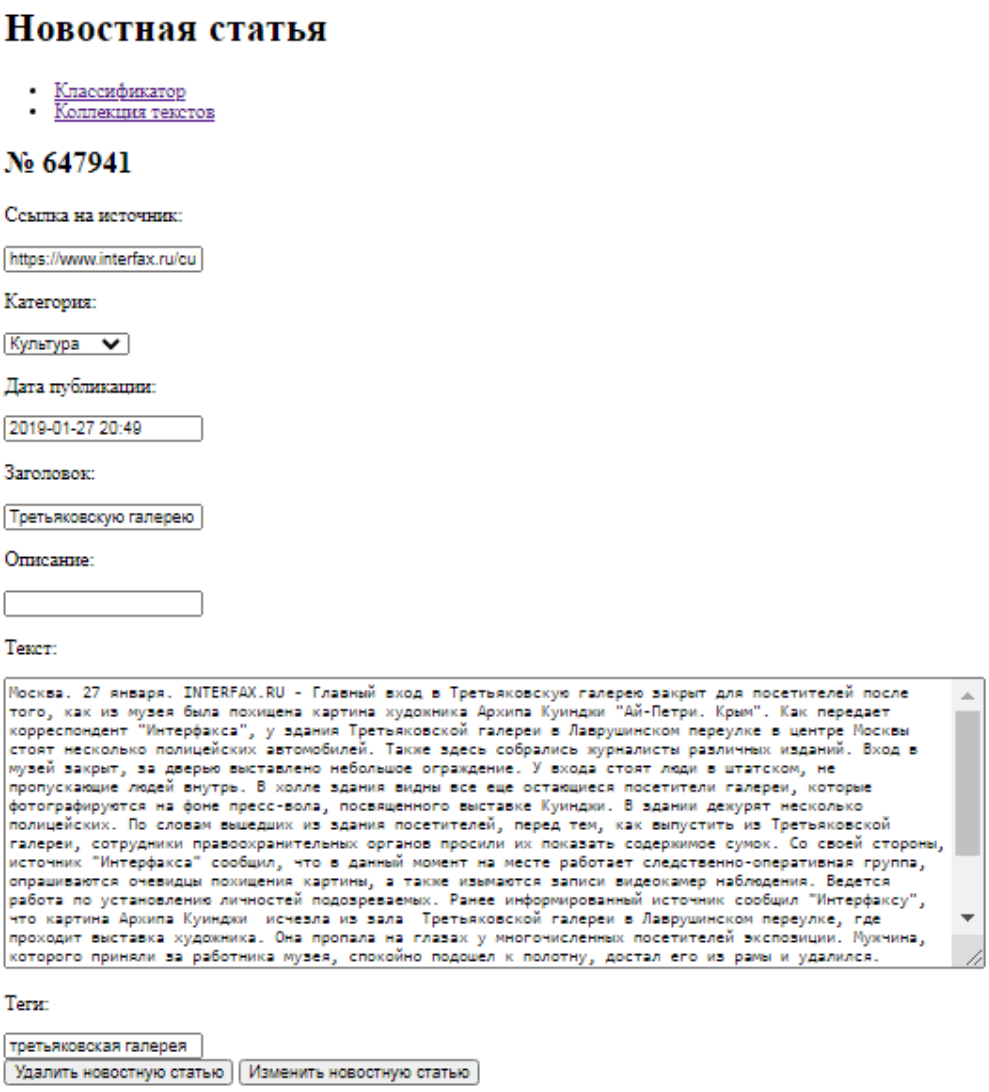

Рисунок 5 – Интерфейс для работы с отдельной новостью

Для добавления новой статьи необходимо ввести данные о ней в форму, затем нажать на кнопку «Добавить статью». Интерфейс для добавления новой статьи представлен на рисунке 6.

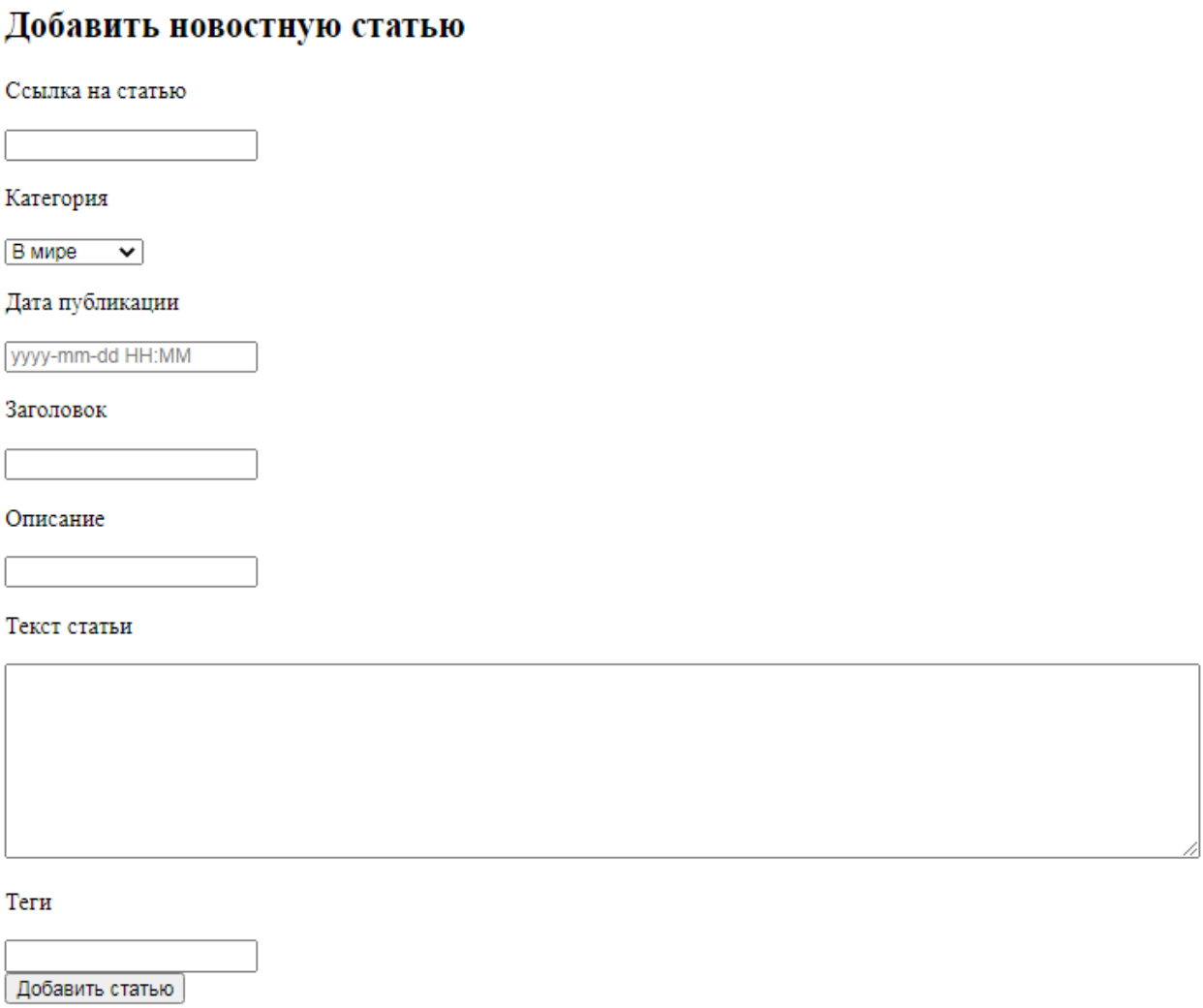

Рисунок 6 – Интерфейс для добавления новой статьи

8 АНАЛИЗ РЕЗУЛЬТАТОВ

Таким образом, был спроектировано и реализовано web-приложение для классификации новостных статей и работы с их коллекцией.

Разработанное web-приложение не требует больших затрат на поддержку, сопровождение и обновление.

К приложению легко добавить новые web-страницы и функционал, однако при изменении уже разработанного функционала требуется приложить много усилий. Также при изменении структуры новостной статьи требуется изменить большое количество программных единиц.

В плане отказоустойчивости, web-приложение защищено от ввода некорректных данных введенных пользователем. Однако при изменении инфраструктуры приложения, а также различных метафайлов возможно нарушение работы приложения.

Классификатор имеют неплохую точность распознавания категорий, а именно 83% процента. Однако надо в дальнейшем повышать данный результат, за счет лучшей подготовки текста. Также у классификатора имеется ещё одна проблема, которая заключается в низкой точности определения категории «В России», что также надо исправлять.

Также frontend разрабатывался в большей степени для браузера Chrome, поэтому могут возникнуть проблемы при использовании приложения на других браузерах.

Необходимо значительно переработать дизайн интерфейса для повышения удобства эксплуатации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Interfax [Электронный ресурс]. – Режим доступа: https://www.interfax.ru/. – Заглавие с экрана. – (Дата обращения: 25.12.20).

2. Python [Электронный ресурс]. – Режим доступа: https://www.python.org/. – Заглавие с экрана. – (Дата обращения: 25.12.20).

3. Django [Электронный ресурс]. – Режим доступа: https://docs.djangoproject.com/. – Заглавие с экрана. – (Дата обращения 25.12.20).

4. Scikit-learn [Электронный ресурс]. – Режим доступа: https://scikit-learn.org/. – Заглавие с экрана. – (Дата обращения 25.12.20).

5. Htmlbook [Электронный ресурс]. – Режим доступа: http://htmlbook.ru/. – Заглавие с экрана. – (Дата обращения 25.12.20).

6. Современный учебник javascript [Электронный ресурс]. – Режим доступа: https://learn.javascript.ru/. – Заглавие с экрана. – (Дата обращения 25.12.20).