Предметом изучения данной статьи является приложение калькулятора с рукописным вводом и его актуальность. Исследованы существующие решения, их достоинства и недостатки. На основе выявленных особенностей готовых решений предложено актуальное решение. Разработана математическая модель, архитектура и сценарии использования предлагаемого решения. Проведены результаты экспериментов для различных методов классификации рукописного текста, а также для параметров, улучшающих работу классификатора на имеющихся данных. Приведены результаты экспериментов, а также предложены варианты дальнейшего развития приложения.

Ключевые слова: калькулятор с рукописным вводом, многоклассовая классификация, метод опорных векторов, MNIST.

**Актуальность решения**

На сегодняшний день гаджеты и техника окружают нас всюду, более того, её «интеллектуальные способности» с каждым годом всё больше удивляют человечество. Ученые и специалисты работают над тем, чтобы каждому пользователю было максимально комфортно «общаться» с техникой: доносить до неё свой запрос и получать нужный результат.

В связи с прогрессом интеллекта у машин никого не удивишь такими средствами общения с компьютерами, как голосовой ввод и ввод рукописного текста. Этот функционал давно стал нормой и удобным инструментом для современного пользователя. Легко предположить, что каждый человек, которому привычны и удобны вышеуказанные средства хотел бы общаться с техникой на этом «языке» в любом инструменте, который может предложить компьютер или какой-либо гаджет.

Практически каждый пользователь с той или иной частотой использует калькулятор. Неважно, является ли пользователь бухгалтером, который практически каждый день сталкивается с расчетами затрат фирмы, или домохозяйкой, которой необходимо подсчитать затраты на коммунальные услуги, или же студентом, который просто хочет подсчитать ответ задачи. В любом случае, каждая целевая аудитория сталкивается с вычислениями.

Из вышеуказанного материала можно сделать вывод, что приложение калькулятора с рукописным или голосовым вводом было бы весьма актуально среди пользователей, которые привыкли пользоваться данными инструментами. Поскольку ввести математическое выражение при помощи голосового ввода – довольно затруднительная задача, стоит рассмотреть вариант создания калькулятора с рукописным вводом.

Особенно актуальным это приложение было бы для людей с дальнозоркостью: люди, страдающие эти недугом, плохо разбирают мелкие надписи, поэтому им было бы очень практично иметь возможность вести записи в более крупном виде, а не всматриваться в каждую кнопку, в поисках нужной буквы или цифры.

**Существующие решения**

Так как калькулятор срукописным вводом удобен и практичен, некоторые компании и разработчики уже создали свои решения данной задачи. На просторах интернета удалось найти следующие инструменты:

1. MyScriplculator

На сегодняшний день этот калькулятор является одним из лучших: он распознает математические выражения целиком, знает множество математических функций, общепринятых обозначений и даже умеет искать недостающие части уравнений.

Существует версия для Android и iOS. Однако те, кому было бы удобно работать на своем ПК с рукописным калькулятором, с легкостью могут решить проблему отсутствия реализации настольной версии при помощи виртуальной машины.

1. Touch-Calculator

Реализация калькулятора с рукописным вводом для системы MacOS. Калькулятор имеет как интерфейс с “кнопочными” цифрами и математическими знаками, так и поле для рукописного ввода. Распознавание проходит достаточно точно и быстро. Её единственный недостаток – распознавание происходит только по одному символу.

1. Web-версия

Работает с довольно большим процентом ошибок при распознавании и приложение неудобно в использовании, производит только распознавание математического выражения без выведения результата вычислений

В таблице 1 представлено сравнение существующих решений.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Качество классифи-кации | Распознава-ние выражения целиком | Вычисление значения | MacOS | Windows | Linux | Android | iOS | Web |
| MyScript Calculator | Высокое | Да | Да | Да | Да | Да | Да | Да | Нет |
| Touch Calculator | Высокое | Нет | Да | Да | Нет | Нет | Нет | Нет | Нет |
| Mathematical Expression Recognition | Низкое | Да | Нет | Нет | Нет | Нет | Нет | Нет | Да |

Таблица 1

Таким образом, существуют удобные и качественные реализации для мобильных устройств, которые можно использовать как настольные версии при помощи виртуальной машины. Очевидно, что самым кроссплатформенным решением является web-версия приложения. У существующего web-приложения калькулятора есть реализация, однако она обладает следующими негативными свойствами: интерфейс не стилизован, низкое качество классификации, не производит вычисление введенного выражения . Из обзора существующих решений задачи можно сделать вывод о том, что самым актуальным решением будет создание web-версии калькулятора, который распознает математическое выражение с высокой точностью и вычисляет его.

**Математическая модель решения**

Создание приложения калькулятора с рукописным вводом состоит из следующих этапов:

* Определение границ каждого из символов(сегментация изображения)
* Определения класса каждого символа
* Парсинг получившейся строки и вычисление результата

Для определения границ символов и парсинга строки в Python есть библиотеки, качество работы которых высоко. В случае с определением класса каждого объекта хоть и существуют готовые реализации методов классификации, немобходимо проводить иследование с подбором параметров для качественной работы методов.

Чтобы новое веб-приложение калькулятора с рукописным вводом оказалось лучшим решением, чем уже существующее, необходимо организовать качественную классификацию рукописных символов.

Были исследованы следующие методы классификации:

* Мультиномиальная логистическая регрессия
* Наивный Байессовский классфикатор
* Метод k ближайших соседей
* Метод Парзеновского окна
* Метод потенциальных функция
* Метод опорных векторов

После подбора наилучших параметров для выборки MNIST получены следующие результаты:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Качество классификации (доля тестовой выборки равна 10%) | Скорость классификации 100 объектов | Скорость обучения | Необходимость хранения выборки |
| Мультиномиальная логистическая регрессия | 92,4% | 0.005 секунд | 110 секунд | Нет |
| Наивный Байессовский классфикатор | 92,44% |  |  | Нет |
| Метод k ближайших соседей | 97,15% | 110 секунд | 110 секунд | Есть |
| Метод Парзеновского окна | 94,14% | 110 секунд | 110 секунд | Есть |
| Метод потенциальных функций |  |  |  | Есть |
| Метод опорных векторов | 98,52% | 0,001 секунда | 1,5 часа | Нет |

Таким образом, наилучшее качество классификации на выборке MNIST показали методы k ближайших соседей и опорных векторов.

Метод k ближайших соседей требует хранения всей выборки, размеры которой приблизительно 80 Мб, а также для каждого нового классифицируемого объекта требуется переобучение классификатора, что сказывается на скорости работы приложения. Для одного символа классификация занимает примерно 1 секунду, то есть выражение в 5 символов займёт приблизительно 5 секунд. Для современного пользователя это недопустимо длительное время ожидания результата.

Метод опорных векторов осуществляет классифкикацию символа всего за 0,001 секунды и не требует хранения данных, только значение полученных параметров при обучении. Данный подход обладает одним недостатком: долгое обучение. Однако обучение требуется только один раз, поэтому после выполенения обучения на сервере пользователи смогут быстро получать результат.

Итак, в качестве метода классификации для приложения калькулятора с рукописным вводом самым оптимальным выбором является метод опорных векторов.

Рассмотрим подробнее, каков его принцип работы.

Метод опорных векторов

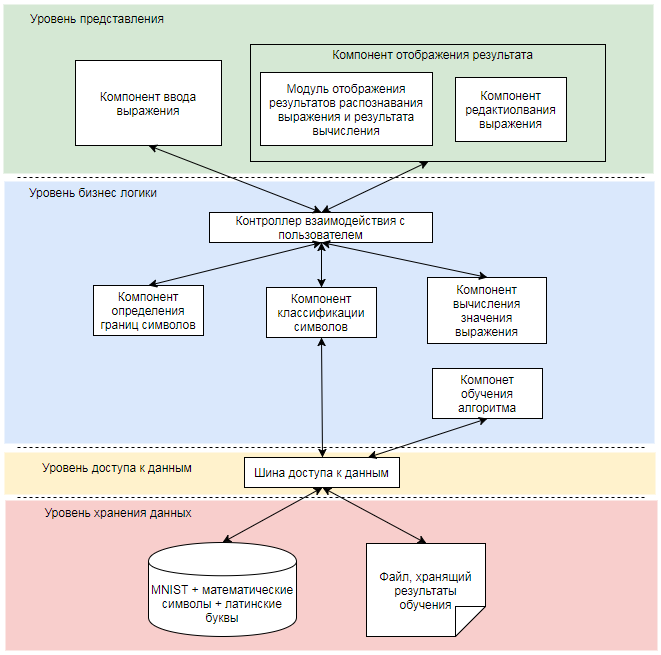
Данный метод первоначально предназначен для бинарной классификации, однако существует и модификация для многоклассовой. Суть метода заключается в том, что все объекты тренировочной выборки представлены в некотором k-мерном пространстве в виде вектора размерности k. Для разделения имеющихся объектов в пространстве используется так называемая плоскость классификатора, которая представляет собой гиперплоскость размерностью k-1. Логично, что таких плоскостей можно провести бесконечно большое количество. В алгоритме опорных векторов лучшей разделяющей плоскостью считается плоскость, расстояние от которой до каждого из классов максимально. В итоге пространство оказывается разделено на участки, каждый из которых соответствует какому-либо классу.

После того как плоскость проведена определяется положение каждого нового объекта ( то есть объекта, для которого необходимо решить задачу классификации) и ему присваивается тот класс, в участок которого попал этот объект.

В Python существует библиотека sklearn, которая содержит в себе реализацию метода метода опорных векторов. Используем её для определения наилучших параметров при классификации выборки MNIST.

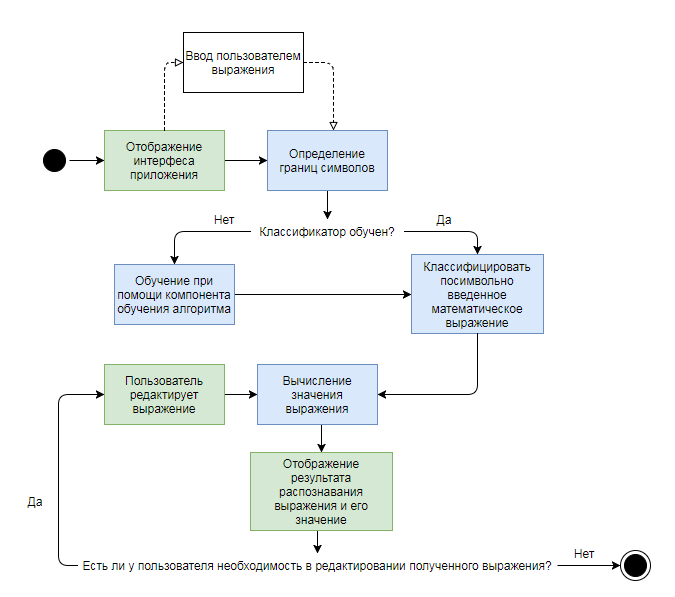
Эксперименты по подбору наилучших параметров для выбранного метода приведены в пункте «Полученные результаты».

**Архитектура**

****

**Сценарии использования**

Работа приложения, используя созданную архитектуру:

****

1. Распознавание выражения
   1. Пользователь заходит в приложение
   2. Пользователь видит поле для ввода выражения, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода»
   3. Пользователь вводит математическое выражение в поле для ввода рукописного текста
   4. Пользователь нажимает кнопку «Вычислить значение выражения»
   5. Пользователь видит под кнопками текстовое поле с полученным после распознавания выражением и его результат
2. Редактирование результата распознавания
   1. Пользователь заходит в приложение
   2. Пользователь видит поле для ввода выражения, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода»
   3. Пользователь вводит математическое выражение в поле для ввода рукописного текста
   4. Пользователь нажимает кнопку «Вычислить значение выражения»
   5. Пользователь видит под кнопками текстовое поле с полученным после распознавания выражением и его результат
   6. Пользователь получает некорректное значение
   7. Пользователь наводит курсор на распознанное выражение и ставит его в месте, где необходимо провести редактирование
   8. Пользователь редактирует выражение и автоматически видит новый ответ
3. Повторное распознавание выражения
   1. Пользователь находится в приложении
   2. Пользователь видит поле для ввода выражения с введённым на нём выражением, кнопку «Вычислить значение выражения», кнопку «Очистить поле для ввода», распознанное предыдущее выражение и его результат
   3. Пользовательно нажимает кнопку «Очистить поле для ввода» и видит чистое поле для ввода
   4. Пользователь вводит математическое выражение в поле для ввода рукописного текста
   5. Пользователь нажимает кнопку «Вычислить значение выражения»
   6. Пользователь видит под кнопками текстовое поле с полученным после распознавания выражением и его результат

**Полученные результаты**

Код всё ещё выпоняется ☹

**Дальнейшее развитие решения**

В первоначальной версии приложения предусмотрены только распознавание строковых выражений, без возведения в степень, извлечения корней, дробей и т.д. При дальнейшем развитии приложения можно добавить больше математических возможностей.