МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ ЗАИМСТВОВАНИЯ
Заведующий кафедрой
д.п.н., профессор
_____И.Г. Захарова
29 июня 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ВЫЯВЛЕНИЯ И АНАЛИЗА СОСТАВА ПРОДУКТОВ НА ОСНОВЕ ИЗОБРАЖЕНИЯ ТОВАРА

02.04.03. Математическое обеспечение и администрирование информационных систем
Магистерская программа «Высокопроизводительные вычислительные системы»

Выполнил работу
Студент 2 курса
очной формы обучения
Борисов Виталий Викторович

Научный руководитель
д. п. н., профессор
Захарова Ирина Гелиевна

Рецензент
к. фил. н., доцент кафедры ИС
ФГБОУ ВО «ТюмГУ»
Бидуля Юлия Владимировна

Тюмень 2018

Оглавление

Введение **3**
Проблематика 3
Цель 4
Задачи 4
Глава 1. Обзор решений задачи определения состава продуктов **5**
Перечень ингредиентов 5
Пищевые добавки 6
Обзор существующих решений 8
Приложение “Ingred” 8
Приложение “Едим - Глядим” 10
Глава 2. Особенности распознавания текста на изображении **12**
Этапы распознавания текста 12
Бинаризация 12
Определение языка 13
Определение ориентации страницы 14
Классификация символов 14
Выделение слов 18
Распознавание текста с помощью Tesseract OCR 21
Предобработка - устранение искажений 23
Постобработка - исправление опечаток 24
Глава 3. Разработка программного обеспечения **26**
Логическая схема базы данных 26
Список технологий 28
Программная реализация 29
Описание интерфейса приложения 33
Заключение **35**
Источники **36**
Приложение А - Отправка фото на сервер **38**
Приложение Б - Обработка HTTP запроса на сервере **39**
Приложение В - Обработка искажений **39**
Приложение Г - Распознавание текста на изображении **40**

Введение

Развитие технологий вносит вклад в популяризацию здорового образа жизни, делая доступным опыт и знания. Супермаркеты предлагают нам большой выбор среди продуктов, которыми люди питаются каждый день. Однако, зачастую покупатели не могут быть уверены, что их ингредиенты не приносят вреда здоровью. Решение о покупке осложняется тем, что один и тот же продукт от двух разных производителей может иметь кардинально разный состав. На помощь приходят специализированные приложения для мобильных устройств, например “Едим - Глядим”, которые используют штрих-код для поиска компонентов. Однако база данных продуктов таких приложений может быть неполной, либо неточной. Существует также другая категория приложений, которые обрабатывают непосредственно состав с этикетки, но отечественных, либо локализованных решений с таким подходом нет.

Проблематика

Покупателю в магазине необходимо оперативно проверить состав продукта на наличие в нем вредных ингредиентов и добавок, используя мобильное приложение. Обзор существующих приложений, в частности, для Android, выявил два вида решений. Они имеют следующие недостатки:

1. Приложение определяет состав по штрихкоду. База данных штрихкодов обновляется не оперативно, также в неё не попадают малоизвестные товары и бренды. Это вносит определенные проблемы, связанные с доступностью информации о продуктах. Приложения используют, как правило, встроенную локально базу данных, это не позволяет получать актуальную информацию.

2. Приложение определяет состав по фото. Русскоязычных приложений с таким функционалом нет.

В связи с этим необходимо создать приложение, которое по фото состава будет выводить информацию о вреде ингредиентов для организма.

Цель

Разработать приложение для автоматического распознавания состава продукта по фотографии и вывода информации о вреде отдельных ингредиентов для организма.

Задачи

- Составить алгоритм определения состава по фотографии. Принять во внимание следующие проблемы:
  - Мелкие буквы
  - Плохой шрифт
  - Плохая камера
  - Блики от упаковки
  - Искривленная поверхность (например, бутылки)
- Собрать базу о свойствах ингредиентов, входящих в состав продуктов
- Организовать систему текстового поиска с опечатками по наименованию ингредиентов, с целью уменьшения уровня ошибок при плохом качестве распознавания текста с изображения
- Разработать мобильное приложение, включающее обработку фотографии и вывод справочной информации о продуктах питания

Глава 1. Обзор решений задачи определения состава продуктов

Предметная область данной работы - состав продуктов питания, продающихся на территории Российской Федерации.

Перечень ингредиентов

На упаковке каждого продукта питания в соответствии с ГОСТ Р 51074-2003 необходимо указывать его перечень ингредиентов. Его приводят для всех пищевых продуктов, за исключением продуктов, состоящих из одного ингредиента. Перед списком ингредиентов должен быть заголовок «Состав». Ингредиенты перечисляют в порядке уменьшения массовой доли в момент изготовления пищевого продукта. Если ингредиент представляет собой пищевой продукт, состоящий из двух или более ингредиентов, то такой составной ингредиент допускается включать в перечень ингредиентов под собственным наименованием. При этом непосредственно после наименования такого составного ингредиента в скобках приводят список составляющих его компонентов в порядке уменьшения их массовой доли.

При указании пищевых добавок используют следующие групповые наименования пищевых добавок: антиокислители; вещества для обработки муки; вещества, препятствующие слеживанию и комкованию; вещества, способствующие сохранению окраски; влагоудерживающие агенты; глазирователи; желеобразователи; загустители; кислоты; консерванты; красители; наполнители; Отвердители; пеногасители; пенообразователи; пропелленты; подсластители; разрыхлители; регуляторы; стабилизаторы; уплотнители; усилители вкуса и запаха; эмульгаторы; эмульгирующие соли.

После группового наименования указывают индекс согласно Международной цифровой системе (INS) или Европейской цифровой

системе (Е), или название пищевой добавки. Для ароматизаторов должно быть указано: «натуральный», «идентичный натуральному» или «искусственный» в зависимости от того, какими они являются. Виды заболеваний, при которых противопоказано применение отдельных видов пищевых продуктов и добавок, определяет Министерство здравоохранения Российской Федерации.

Пищевые добавки

Согласно Европейской цифровой кодификации пищевые добавки подразделяют следующим образом:

- Е-100 —Е-182 —красители;
- Е-200 — Е-299 — консерванты;
- Е-300 —Е-399 — антиокислители (антиоксиданты);
- Е-400 —Е-449 — стабилизаторы консистенции;
- Е-450 — Е-499 — эмульгаторы;
- Е-500 — Е-599 — регуляторы кислотности, разрыхлители;
- Е-600 — Е-699 — усилители вкуса и аромата;
- Е-700 — Е-800 — запасные индексы для другой возможной информации;
- Е-900 и далее — антифламинги, улучшители качества хлеба и т.д.

Основными документами, регламентирующими применение пищевых добавок в продуктах на территории Российской Федерации являются:

- Федеральный закон «О санитарно-эпидемиологическом благополучии населения» от 30.03.1999 г. N 52-ФЗ
- Федеральный закон «О качестве и безопасности пищевых продуктов» от 02.01.2000, N 29-ФЗ
- Федеральный закон «Основы законодательства Российской Федерации об охране здоровья граждан» от 22.07.1993

- СанПиН 2.3.2.1293-03 «Гигиенические требования по применению пищевых добавок» — с 12 июня 2003 года

В частности, список разрешённых к применению добавок перечислен в СанПиН 2.3.2.1293-03.

В других странах существуют аналогичные списки разрешенных добавок, регламентируемые государством. Так например, в европейском союзе существует регламент Европейского Парламента и Совета Европейского Союза 1333/2008 от 16 декабря 2008 г. "О пищевых добавках".

Степени безопасности пищевых добавок:

- Пищевая добавка запрещена к использованию в кормах для животных и пищевых продуктах для человека
- Пищевая добавка полностью безопасна
- Пищевая добавка небезопасна для определенной категории лиц (младенцев, маленьких детей, беременных женщин, лиц старшего возраста и др.)
- Пищевая добавка может вызвать побочные явления, не представляющие угрозу жизни
- Пищевая добавка может вызвать побочные явления, представляющие угрозу жизни (рак, сильные кровотечения и другие), но не запрещена к использованию в пищевых продуктах для человека на территории Российской Федерации
- Пищевая добавка запрещена к использованию в пищевых продуктах для человека, так как вызывает побочные явления, представляющие угрозу жизни

Обзор существующих решений

Приложение “Ingred”

Разработано Zorrosoft[8]. Приложение умеет распознавать состав по фото, при этом показывает хорошую точность распознавания русскоязычного текста и высокую скорость работы. Результат распознавания изображения с рисунка 1:

ТОРТ БИСКВИТНЫЙ-НЕГР В ПЕНЕ» 0 СЕРИИ ПАКОМЫЙ КУСОЧЕК» СОСТАВ ПРОДУКТА: КРЕМ НА РАСТИТЕЛЬНЫХ МАСЛАХ РАСТИТЕЛЬНЫЙ ЖИР САХАР МОЛОЧНЫЙ БЕЛОК ЗМУЛЬГАТОР-ЛЕЦИТИН СТАБИЛИЗАТОРЫ Е340 Е460 Е466 СОЛЬ ПОВАРЕННАЯ ПИЩЕВАЯ ПИЩЕВОЙ КРАСИТЕЛЬ-БЕТА-КАРОТИН МУКА ПШЕ- НИ МАССА НЕТТО: 0 ə ЧНАЯ ХЛЕБОПЕКАРНАЯ В/С ЯЙЦЕПРОДУКТЫ СМЕСЬ ДЛЯ ЗАВАРНОО КРЕМА САХАР ЗАУСТИТЕЛИ Е 1414 Е401 МОЛОКО СООЕ ЦЕЛЬНОЕ СТАБИЛИЗАТОР Е339 ТАЗ Ь КОНДИТЕРСКАЯ ЗАМЕНИТЕЛЬ МАСЛА КАКАО САХАР КАКАО-ПОРОШОК ЗМУЛЬ А ТОР-ПЕЦИТИН САХАР МОПОКО Ц МОЛОКО ОБЕЗЖИРЕННОЕ САХАР РАЗРЫХЛИТЕЛЬ-Т500 ЕЛЬНОЕ СУЩЕННОЕ С САХАРОМ МОЛОКО ЦЕЛЬНОЕ РОДУКТ МОЖЕТ СОДЕРЖАТЬ СЛЕДЫ АРАХИСА RPER ORO ОРЕХА ФУНДУКА МИНДА ПИИЕВАЯ ЦЕННОСТЬ 100 ХРАНИТЬ ПРИ Т-4 2°С ТУ 9130-001-01 54862665-2008 -ЖИРОВ-9 0 R -УТЕВОДОВ-31 0 R

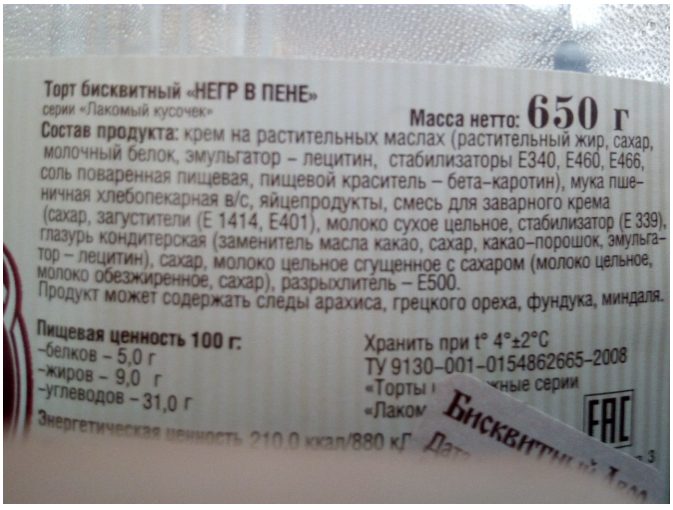

Рисунок 1 - тестовое изображение

Однако приложение не содержит русского словаря со списком ингредиентов и распознает в тексте только добавки, описанные с помощью Е - нотации. Результат работы приложения с изображения на рисунке 1 представлен на рисунке 2:

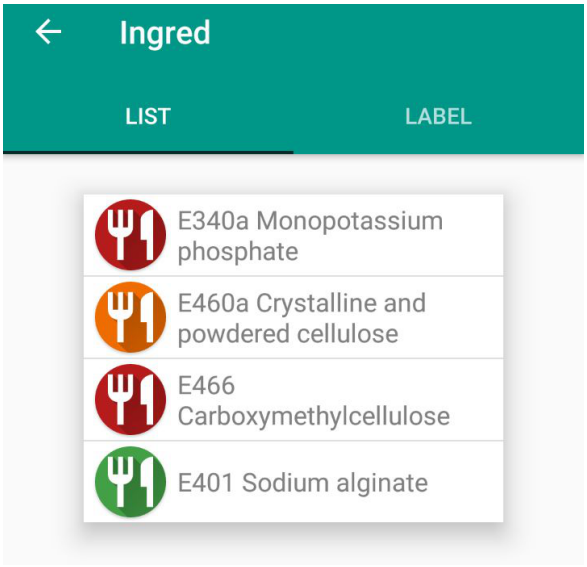

Рисунок 2 - результат работы приложения

По каждой распознанной добавке приложение содержит её характеристику, в которой содержится описание потенциального вреда для здоровья.

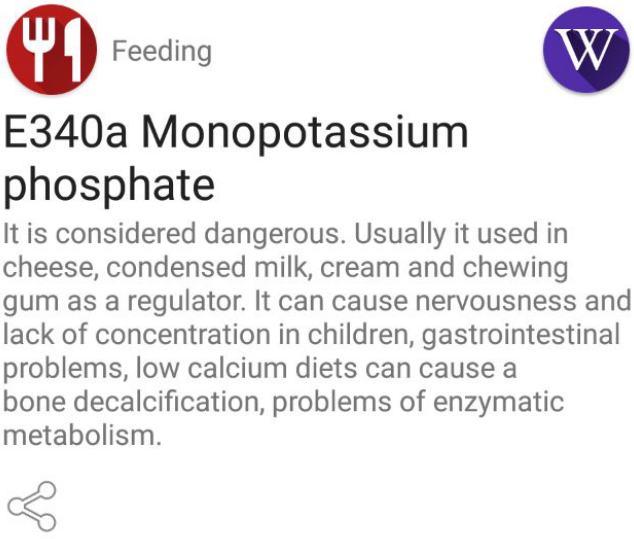

Рисунок 3 - описание добавки

В целом приложение полностью удовлетворяет потребностям пользователя, за исключением отсутствия поддержки русского языка.

Приложение “Едим - Глядим”

Приложение разработано Serg Development[9]. Предоставляет возможность искать товары с помощью штрихкода. Доступно два режима поиска - ручной ввод и распознавание с помощью камеры мобильного устройства. База товаров неполная - так например отсутствует торт, представленный на рисунке 1. Состав имеющихся товаров в приложении не всегда совпадает с фактическим (Рисунок 4). Так например, для сыра “Омичка” в списке отсутствовал “Антиокислитель концентрат токоферолов”. Для опасных добавок доступно подробное описание (Рисунок 5).

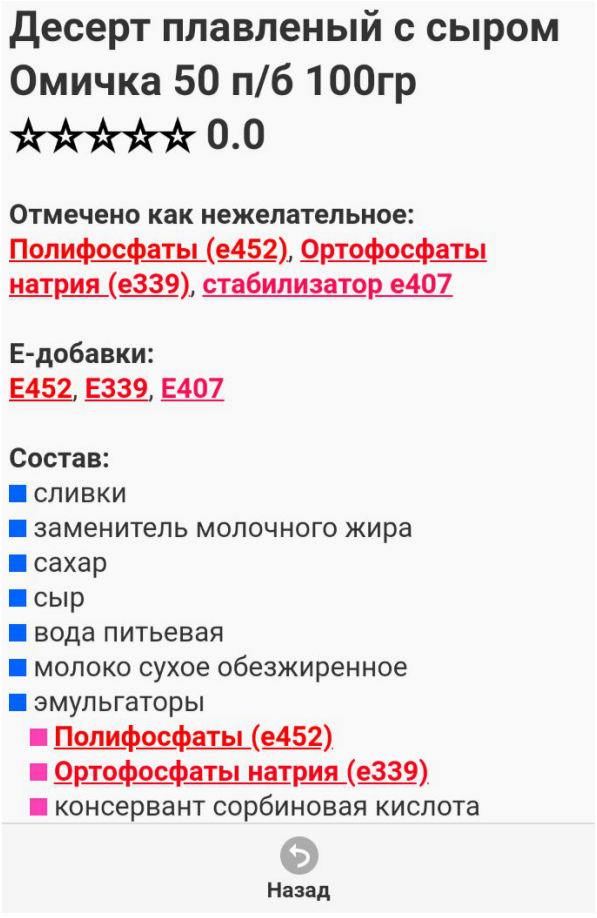

Рисунок 4 - пример работы приложения

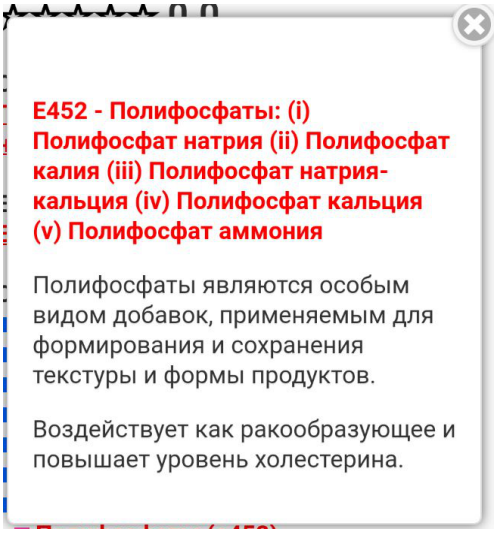

Рисунок 5 - подробное описание добавки

Резюмируя, можно заключить, что хотя приложение содержит недостаточно полную и точную базу продуктов, однако все равно является полезным.

Глава 2. Особенности распознавания текста на изображении

В силу специфики предметной области распознаваемый текст обладает следующими характеристиками:

- буквы печатные
- как правило отсутствуют шумы печати, в частности, непропечатка (разрывы слитных черт символов), "слипание" соседних символов, пятна и ложными точками на фоне вблизи символов и т. п.;
- отсутствует смещение символов или частей символов относительно их ожидаемого положения в строке;
- отсутствует изменение наклона символов;
- высокая вероятность эффектов освещения (теней, бликов и т. п.) из-за съемок камерой в помещениях магазинов
- возможно искажение поверхности из-за формы упаковки продукта

Перечисленные наблюдения необходимо учитывать при решении задачи распознавания символов в данном приложении.

Этапы распознавания текста

Бинаризация

Данный этап заключается в преобразовании цветного изображения в монохромное, в котором присутствует только два вида пикселей - белые (фон) и черные (текст). Для этого существуют различные критерии бинаризации, например, Отсу, Бернсена, Эйквеля, Ниблэка и т.п. Самым эффективным, как по быстродействию, так и по качеству, считается критерий Отсу.

Метод использует гистограмму распределения значений яркости пикселей растрового изображения. Яркость пикселя вычисляется как 0.299R + 0.587G + 0.114B.

Строится гистограмма по значениям $p_i = n_i/N$, где N – это общее кол-во пикселей на изображении, n_i – это кол-во пикселей с уровнем яркости i. Диапазон яркостей делится на два класса с помощью порогового значения уровня яркости k, k — целое значение от 0 до L. Каждому классу соответствуют относительные частоты ω_0 и ω_1:

$$\omega_0(k) = \sum_{i=1}^{k} p_i \quad \omega_1(k) = \sum_{i=k+1}^{L} p_i = 1 - \omega_0(k)$$

Средние уровни для каждого из двух классов изображения:

$$\mu_0(k) = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} \quad \mu_1(k) = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1}$$

Далее вычисляется максимальное значение оценки качества разделения изображения на две части (межклассовая дисперсия):

$$\sigma_{кл} = \omega_0\omega_1(\mu_1 - \mu_0)^2$$

Находится такое k, при котором межклассовая дисперсия максимальна. Все пиксели с яркостью ниже порога становятся белыми, выше - черными. На рисунке 6 представлен пример работы алгоритма.

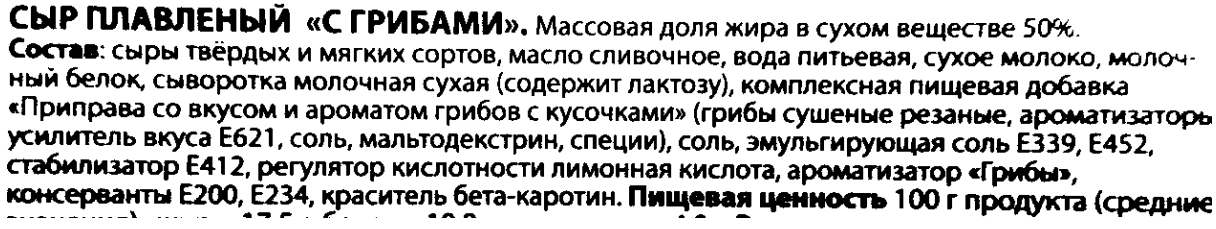
СЫР ПЛАВЛЕНЫЙ «С ГРИБАМИ». Массовая доля жира в сухом веществе 50%.
Состав: сыры твёрдых и мягких сортов, масло сливочное, вода питьевая, сухое молоко, молочный белок, сыворотка молочная сухая (содержит лактозу), комплексная пищевая добавка «Приправа со вкусом и ароматом грибов с кусочками» (грибы сушеные резаные, ароматизаторы усилитель вкуса Е621, соль, мальтодекстрин, специи), соль, эмульгирующая соль Е339, Е452, стабилизатор Е412, регулятор кислотности лимонная кислота, ароматизатор «Грибы», консерванты Е200, Е234, краситель бета-каротин. Пищевая ценность 100 г продукта (средние

Рисунок 6 - результат работы алгоритма бинаризации

Определение языка

Существует три основных подхода к определению языка:

1. “Глобальный”. Алгоритмы данной категории вычисляют отличительные признаки блоков текста для выделения шаблонов, уникальных для конкретного языка или алфавита. У этого подхода есть недостаток в том, что он требует объемного и выровненного

региона текста, а также в том, что признаки чувствительны к шуму или скосу текста.

2. “Локальный”. В нём используются форма и характеристики отдельных компонент связности.
3. “Текстовый”. Алгоритмы в этой категории обрабатывают всю страницу целиком, используя язык по-умолчанию, а затем используют отдельную процедуру для анализа статистики на (потенциально неверном) ответе для того, чтобы угадать, на каком языке на самом деле был написан текст. Алгоритм опирается на то, что на выводе будет мусорный текст, но с характерными частотами символов конкретного языка. Однако такой подход не показывает себя хорошо на большом количестве языков.

Определение ориентации страницы

Зачастую данный этап комбинируется с предыдущим. Изображение поворачивается на 90, 180, 270 градусов. Затем из четырёх вариантов работы алгоритма выбирается тот, в котором текст распознался с большей степенью уверенности

Классификация символов

Существуют множество подходов к определению символа, изображенного на картинке.

Далее перечисляются некоторые из них.

1. Метод k ближайших соседей

Каждое изображение символа характеризуется вектором в пространстве признаков, а также непосредственном классом - нарисованному символу.

Для примера, пространством признаков могут являться цвета пикселей фиксированного по ширине и высоте изображения.

Эталонные изображения символов переводятся в такие вектора, в дальнейшем по ним будет осуществляться поиск.

При поступлении на вход изображения для распознавания алгоритм переводит его в вектор и находит ближайшее по расстоянию изображение среди эталонных изображений, либо k ближайших изображений, и среди них выделяется самый распространенный класс символа. Визуальная интерпретация алгоритма представлена на рисунке 7.

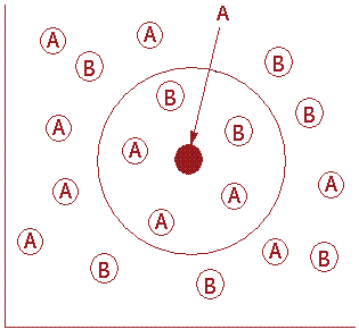

Рисунок 7 - метод k ближайших соседей

2. Метод опорных векторов

Также, как и в методе ближайших соседей, каждое изображение символа характеризуется вектором в пространстве признаков и классом.

В пространстве признаков вычисляется гиперплоскость, которая делит пространство признаков так, что расстояние от нее до каждого класса максимально (разделяет классы наилучшим образом).

Чтобы отнести изображение к какому-либо классу, вычисляется, с какой стороны от гиперплоскости лежит вектор распознаваемого изображения. Класс назначается в зависимости от региона, к которому был отнесен вектор. Визуальная интерпретация алгоритма представлена на рисунке 8.

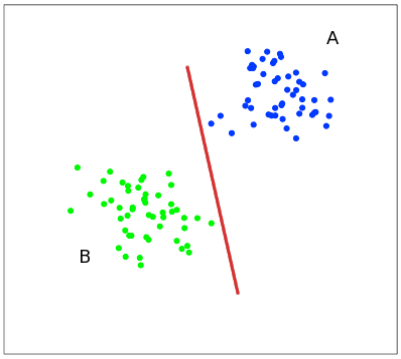

Рисунок 8 - метод опорных векторов

3. Нейросети

Для классификации могут использоваться, например, свёрточные нейронные сети или глубокие сети доверия. На вход нейросети подаются также цвета пикселей фиксированного по ширине и высоте изображения. Особенность нейросетей в том, что они в процессе обучения выявляют характерные признаки каждого класса символов.

4. Классификация на основе контуров

Подход, используемый в библиотеке Tesseract, заключается в следующем:

На изображении выделяются контуры. Затем контур делится на признаки - трёхмерные вектора фиксированной длины (координата x, координата y, градус, показывающий направление) (Рисунок 9). Все компоненты вектора имеют размерность от 0 до 255.

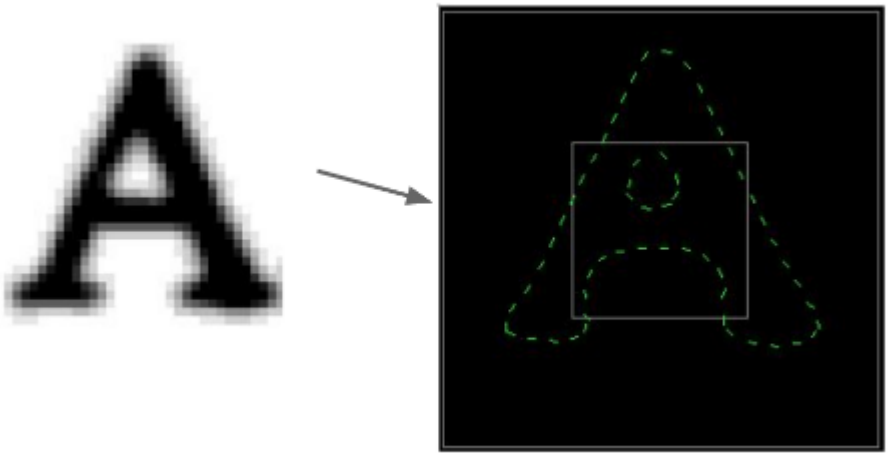

Рисунок 9 - представление изображения в виде векторов фиксированной длины.

Эталонные вектора являются четырёхмерными (Отсюда название Tesseract). В дополнение к трём размерностям, указанным выше, добавляется длина вектора.

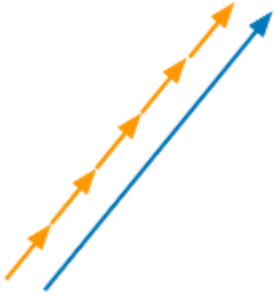

Рисунок 10 - эталонный вектор и вектора классифицируемого изображения

Расстояние между признаком распознаваемого изображения и эталоном вычисляется с помощью длины перпендикуляра между векторами и угла между ними следующим образом: расстояние^2 + угол^2 (Рисунок 11).

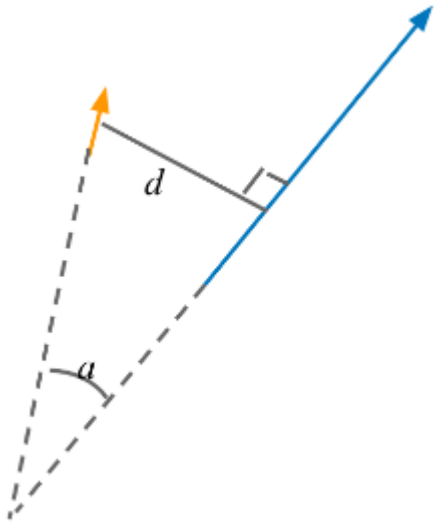

Рисунок 11 - эталонный вектор и вектора классифицируемого изображения

С помощью метода k ближайших соседей для каждого признака строится список подходящих классов (с весами). Затем по всем признакам выбирается наиболее подходящий класс (Рисунок 12).

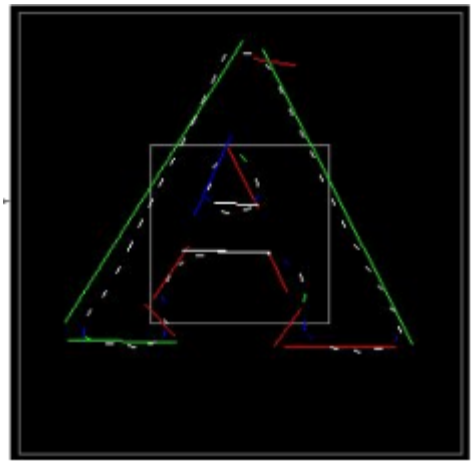

Рисунок 12 - сопоставление с эталоном

Выделение слов

Первоначально выполняется выделение связанных компонент. Алгоритм работы следующий: для каждого черного пикселя на изображении, слева направо, сверху вниз, проверяется:

1. Если соседние с ним черные пиксели не были пронумерованы, то ему даётся номер, на один больший числа уже помеченных компонент
2. Если есть соседний пиксель, который был пронумерован, то текущему пикселю даётся номер соседнего пикселя
3. Если у соседних пикселей два разных номера, то выбирается наименьший номер, и все пиксели, которые имеют оставшийся больший номер, получают меньший номер.

В результате работы каждому выданному номеру соответствует свой связанный компонент.

Выделение слов производится на основе результатов работы алгоритма нахождения связанных компонент и классификатора символов (Рисунки 13, 14).

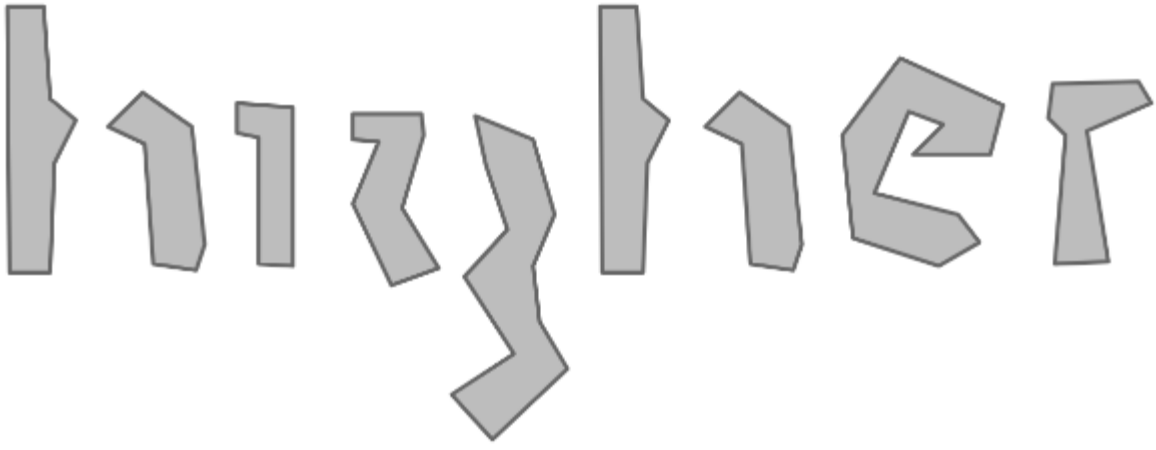

Рисунок 13 - результат работы алгоритма нахождения связанных компонент

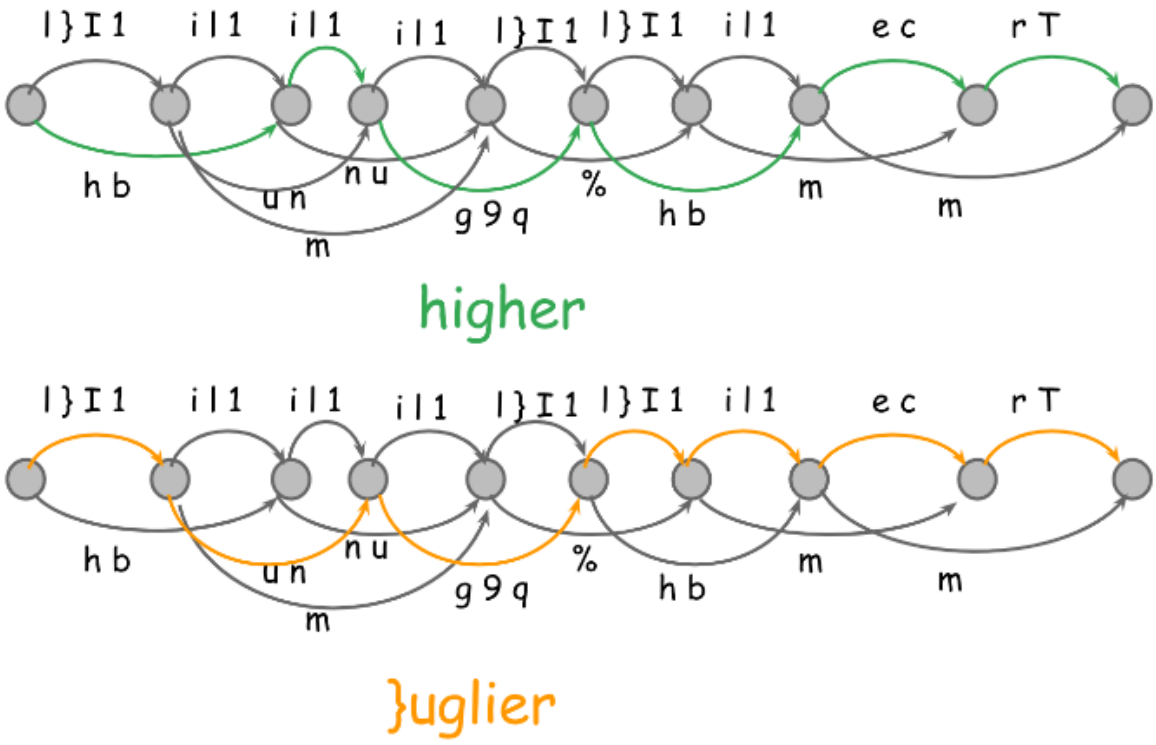

Рисунок 14 - промежуточный результат работы алгоритма по выделению слов

По возможным вариантам группирования связанных компонентов и результатам работы классификатора символов строится ориентированный граф подходящих слов.

Далее, выявленные слова-кандидаты ищутся по встроенному словарю языка и сортируются в порядке увеличения средней ошибки классификаторов символов. Конечным результатом становится слово с наименьшей ошибкой.

Распознавание текста с помощью Tesseract OCR

Tesseract OCR - кроссплатформенная бесплатная библиотека для компьютерного зрения с поддержкой более 100 языков. Она работает с этапами обнаружения, выделения и распознавания символов. Библиотека способна полностью работать с исходными изображениями без необходимости их предварительной обработки сторонними утилитами. Для выделения текста использует сети долгой краткосрочной памяти.

Поддерживает множество режимов работы, среди них:

- Только определение ориентации и языка текста
- Автоматическая сегментация страницы, без определения ориентации и языка. (Режим по-умолчанию)
- Изображение содержит один цельный блок текста
- Изображение содержит одну строку
- Изображение содержит одно слово

В рамках данной работы выявилось, что tesseract плохо справляется с изображениями, на которых помимо состава продукта присутствует много лишних деталей (Рисунок 15).

Рисунок 15 - фото состава продукта

Гораздо лучше он справляется с распознаванием, если передать ему только часть изображения, содержащую состав продукта.

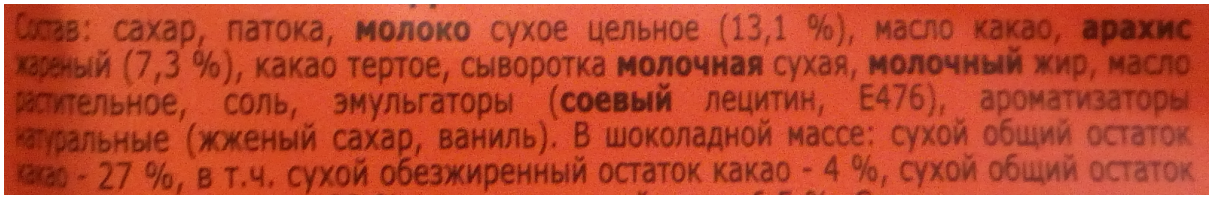

Рисунок 16 - увеличенный состав продукта

Результат распознавания данного изображения с помощью tesseract:

ШЖ2в: сахар, патока, молоко сухое цельное (13,1 %), масло какао, арахис)
Жарный (7,3 %), какао тертое, сыворотка молочная сухая, молочный жир, масло_
апительное, соль, эмульгаторы (соевый лецитин, Е476), ароматизаторы”
Юпальные (жженый сахар, ваниль). В шоколадной массе; сухой общий остаток
20 - 27 %, в т.ч, сухой обезжиренный остаток какао - 4 %, сухой общий остаток

Также анализировалось качество распознавания для искривленных поверхностей. Выявлено, что Tesseract плохо обрабатывает такие изображения, как например на рисунке 17:

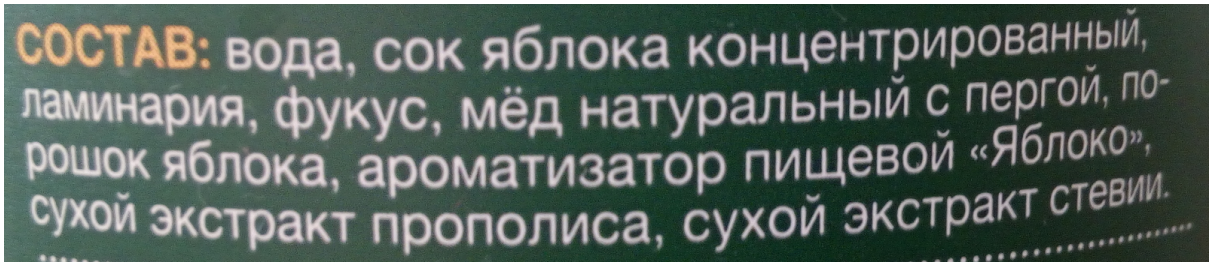

Рисунок 17 - поверхность банки

Результат работы tesseract на рисунке номер 17:
9 об сок яблока концентрированный
,ламинария, фукус, мёд натуральный с иТе еде ДА я
нок неожы ароматизатор ВУИ ет орал ЛК
")^ой Экстракт прополиса, орет ео

Таким образом, для повышения качества работы tesseract необходима пред- и постобработка фотографий.

Предобработка - устранение искажений

Для устранения искажений, вызванных выпуклой формой упаковок продуктов питания, была выбрана утилита Document Image Dewarping. В основе её работы лежит анализ не только строк текста, но ещё и отрезков прямых. В основе алгоритма лежит предположение, что отрезки в реальности не искривлены, а также направлены вертикально, либо горизонтально. Влияние предложенного изменения можно увидеть на рисунке 18.

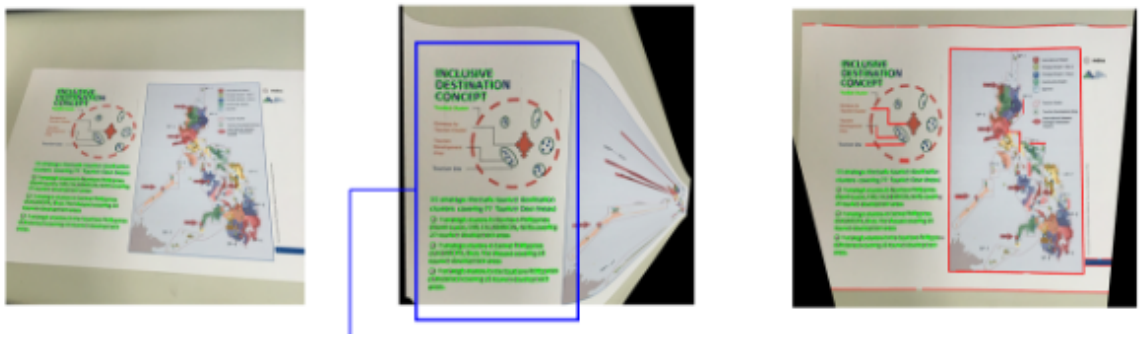

Рисунок 18 - слева оригинальное изображение, в центре устранение искажений с использованием только строк текста, справа предложенный метод

Алгоритм работает в итеративной манере, оптимизируя целевую функцию (Формула 1). В формуле 1 Θ это параметры алгоритма, подлежащие оптимизации; f_{text} оценивает искривления строк текста. Данная оценка мала, когда строки выровнены по горизонтали, а межстрочный интервал между парами соседних строк одинаков. f_{line} - оценка искривления отрезков. Коэффициент λ задается как пропорция между количеством строк текста и количеством отрезков на изображении.

$$f_{cost}(\Theta) = f_{text}(\Theta) + \lambda f_{line}(\Theta)$$

Формула 1 - целевая функция

Для оценки искривления отрезков используется формула 2, где θ_i означает угол i-го отрезка относительно горизонтальной прямой.

$$f_{line} = \sum_i min(cos^2\theta_i, sin^2\theta_i)$$

Формула 2 - целевая функция для отрезков

В алгоритме учтено, что не все отрезки на изображении в действительности направлены горизонтально или вертикально. Для этого после каждой итерации алгоритма из рассматриваемых отрезков исключаются те, у которых оценка f_{line} осталось большой.

Результат работы алгоритма с рисунком 17 изображен на рисунке 19:

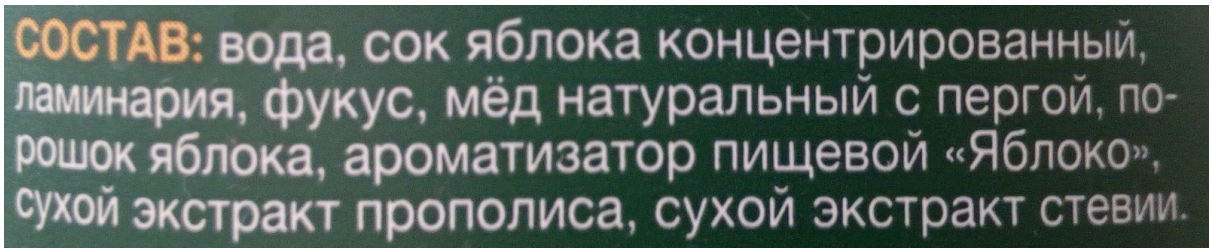

Рисунок 19 - результат работы утилиты

После такой предобработки качество распознаваемого tesseract текста заметно повысилось:

СОСТАВ: вода, сок яблока концентрированный,
ламинария, фукус, мёд натуральный с пергой, по-
Рошок яблока, ароматизатор пищевой «Яблоко».
сухой экстракт прополиса, сухой экстракт стевии.

Постобработка - исправление опечаток

В случае, когда система распознавания не смогла определить часть букв в слове, можно исправить опечатки с помощью нечеткого поиска по заранее составленному словарю ингредиентов. Под нечетким поиском подразумевается задача нахождения по заданному слову слова, имеющего наименьшее количество различий с заданным. Алгоритмы нечеткого

поиска характеризуются метрикой — функцией расстояния между двумя словами, позволяющей оценить степень их сходства в данном контексте. Наиболее распространенные метрики - расстояния Левенштейна и Дамерау-Левенштейна.

Расстояние Левенштейна - это минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Расстояние Дамерау — Левенштейна - модификация расстояния Левенштейна: к 3 операциям добавлена операция транспозиции (перестановки двух соседних символов).

Одной из реализаций поиска с опечатками с применением данного расстояния является SymSpell.

Алгоритм SymSpell хранит словарь всех слов, а также всех возможных слов с опечатками с заданным максимальным редакционным расстоянием. При этом в качестве опечаток выступает только операция удаления. Т.к. операции добавления, изменения, перемещения символа не хранятся, при поиске в словаре выполняются четыре быстрых вида сравнения:

1. Словарное слово = Входное слово
2. Удаление(Словарное слово) = Входное слово
3. Словарное слово = Удаление(Входное слово)
4. Удаление(Словарное слово) = Удаление(Входное слово)

Скорость работы достигается предварительным вычислением всех возможных опечаток и хранением их в словаре с константным временем поиска. При этом размер словаря остается приемлемого размера, так как хранятся только опечатки, связанные с удалением.

Глава 3. Разработка программного обеспечения

Логическая схема базы данных

По результатам анализа предметной области была спроектирована логическая схема базы данных.

Таблица “Пищевая добавка” – хранит сведения о пищевых добавках, входящих в состав продуктов питания.

Ключ	Название	Тип	По умолчанию	Описание
PK	ID добавки	Integer	NOT NULL	Уникальный идентификатор добавки
	Номер добавки	Text	NOT NULL	Индекс добавки согласно Европейской цифровой системе
	Степень безопасности	Enum	NOT NULL	Один из возможных вариантов “Степени безопасности пищевых добавок” (см. описание предметной области)
	Название	Text	NOT NULL	Название пищевой добавки, включая синонимы, через запятую
	Описание	Text	NOT NULL	Подробная информация о

				побочных эффектах добавки

Таблица “Функция” – хранит возможные варианты функций пищевых добавок, согласно Санитарно-эпидемиологическим Правилам и Нормативам 2.3.2.1293-03 (Например, краситель, консервант, стабилизатор, регулятор кислотности и т.п.).

Ключ	Название	Тип	По умолчанию	Описание
PK	ID функции	Integer	NOT NULL	Уникальный идентификатор функции
	Название	Text	NOT NULL	Название согласно перечню 2.3.2.1293-03

Таблица “Функция пищевой добавки” – хранит функции конкретной пищевой добавки.

Ключ	Название	Тип	По умолчанию	Описание
PK	ID добавки	Integer	NOT NULL	Уникальный идентификатор добавки
PK	ID функции	Integer	NOT NULL	Уникальный идентификатор функции

На рисунке 20 изображена логическая схема базы данных.

Рисунок 20. Логическая схема базы данных

Список технологий

Используемые языки: Java для интерфейса, Python для обработки изображения

Используемые редакторы: Android Studio и Atom

Используемые библиотеки Python:

Django - Веб фреймворк для создания HTTP API

Pillow - библиотека для удобной работы с изображениями

Pytesseract - обёртка над библиотекой tesseract

SymSpell - алгоритм исправления опечаток на основе расстояния Дамерау — Левенштейна

SQLite - встраиваемая СУБД

Wine - ПО для запуска приложений, написанных для Windows, на других операционных системах

Subprocess - библиотека для создания и управления процессами операционной системы

Используемые библиотеки Java:

Android-Image-Cropper - Графический компонент для возможности выбора области на изображении

OkHttp - библиотека для выполнения HTTP запросов.

Программная реализация

На рисунке 21 представлена общая схема работы приложения.

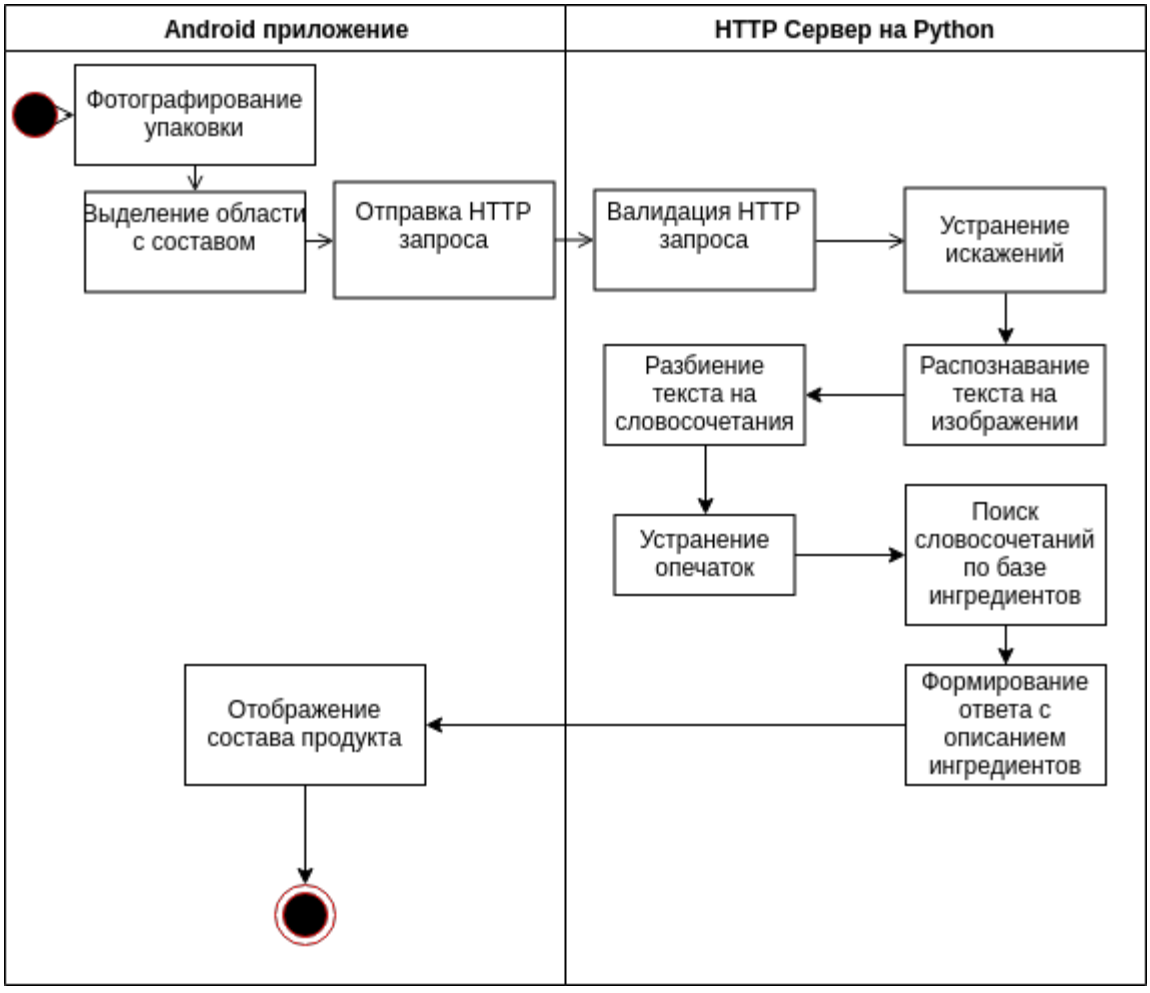

Рисунок 21 - общая схема работы приложения

Для уменьшения нагрузки на телефон пользователя была выбрана клиент-серверная архитектура приложения.

На мобильном телефоне (клиенте) делается снимок, а также выделяется область, которая будет подлежать распознаванию. Опционально изображение может быть повернуто. Общение с сервером происходит по протоколу HTTP, запрос отправляется в отдельном программном потоке, чтобы не блокировать работу интерфейса.

Клиентское приложение отправляет снятое изображение, а также выбранный пользователем регион и поворот в градусах. Такой выбор

обусловлен тем, что алгоритм устранения искажений лучше работает, когда помимо текста вокруг него содержатся элементы оформления этикетки. Программный код отправки запроса представлен в приложении А. В представленном коде создается отдельный программный поток, файл и дополнительные параметры кодируются для передачи по сети. После получения ответа он отображается в приложении в виде списка.

Серверное приложение представляет собой REST api, реализованное на языке Python. Для обработки HTTP запросов был выбрана библиотека Django, т.к. она проста и универсальна. Программный код для обработки запроса клиента представлен в приложении Б. В представленном коде выполняется проверка на наличие в запросе требуемых для дальнейшей работы параметров, вызываются функции распознавания текста, получения списка известных приложению ингредиентов, а также формирование и отправка ответа вызывающему приложению.

Перед началом распознавания переданное от пользователя изображение поворачивается и сохраняется на диск.

Для уменьшения нагрузки на жёсткий диск компьютера и ускорения общей работы системы, промежуточные изображения помещаются в директорию, которая настроена на хранение данных в оперативной памяти.

Для устранений искажений используется утилита Document Image Dewarping, которая представлена в формате исполняемого файла под OS Windows. Серверная часть приложения находится на компьютере под управлением ОС Linux, для работы утилиты была использована программа wine. Программный код для функции устранения искажений представлен в приложении В.

После обработки искажений изображение обрезается в указанном пользователем регионе, чтобы осталась только область с текстом.

Далее, с помощью библиотеки tesseract происходит бинаризация изображения, выделение групп текста, и распознавание отдельных слов и символов в группах. Файлы изображений, созданные во время обработки запроса, удаляются. В приложении Г представлен код, который поворачивает исходное изображение, вызывает функцию устранения искажений, обрезает получившееся изображение по координатам пользователя, вызывает функцию распознавания текста на изображении, затем удаляет изображения с диска и возвращает распознанный текст.

Полученный текст разбивается по знакам препинания и символам-разделителям на слова и словосочетания. Далее каждое слово или словосочетание корректируется по регулярным выражениям и с помощью исправления опечаток нечётким поиском с использованием библиотеки SymSpell. В приложении Д представлены следующие функции:

extract - принимает распознанный текст, вызывает остальные функции для работы над текстом, и возвращает список идентификаторов распознанных ингредиентов

strip_start - удаляет весь текст перед словом “Состав”, для более удобного анализа текста

strip_end - удаляет весь текст после слов, которыми оканчивается описание состава, для более удобного анализа текста

remove_newlines - удаляет переносы строк

remove_meaningless_phrases - удаляет слова, не несущие смысловой нагрузки

split_by_phrases - разбивает текст на отдельные фразы

extract_known_ingredient - ищет в заданной фразе известный ингредиент

Для хранения справочника ингредиентов используется встраиваемая СУБД SQLite. Её применение обусловлено легкостью развертывания и возможностью будущего переноса справочника внутрь приложения.

После выделения отдельных словосочетаний они ищутся по справочнику ингредиентов.

Найденное описание ингредиентов затем отправляется клиенту в формате JSON.

Клиентское приложение отображает полученную информацию в виде списка. При нажатии на элемент списка открывается окно с подробным описанием ингредиента.

Описание интерфейса приложения

Приложение реализовано на платформе Android и состоит из трёх экранов.

При запуске приложения пользователю предлагается снять фото и выделить участок, на котором присутствует состав (Рисунок 22).

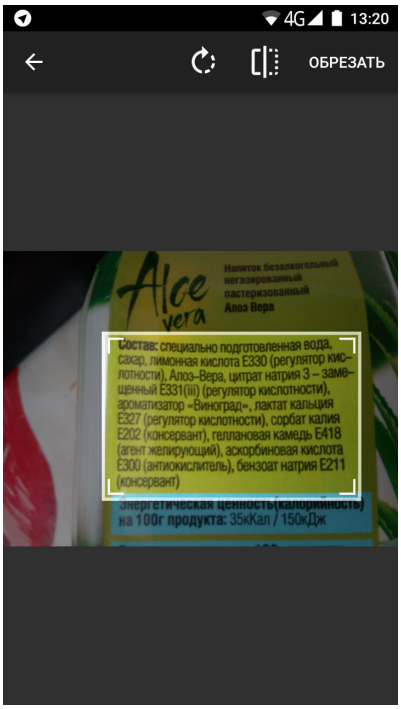

Рисунок 22 - интерфейс выделения состава

После выделения участка изображение загружается на сервер, где и происходит распознавание. В результате пользователю отображается список распознанных ингредиентов, по которым есть информация в базе (Рисунок 23).

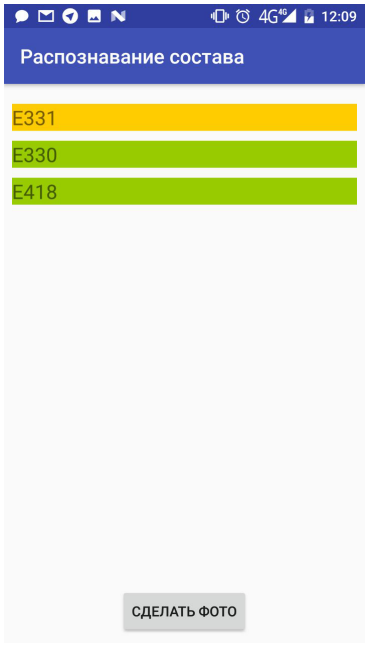

Рисунок 23 - список распознанных ингредиентов

Пользователь имеет возможность посмотреть подробное описание каждого пункта (Рисунок 24), а также сделать новое фото.

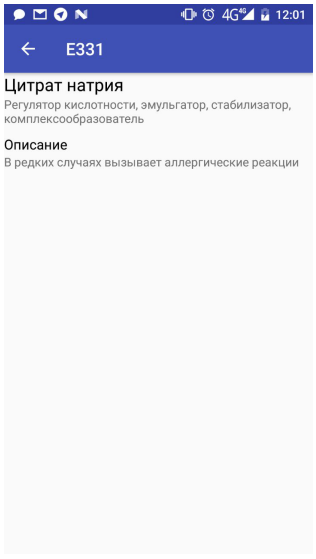

Рисунок 24 - подробное описание состава

Заключение

В рамках выполнения работы были изучены нормативные документы, классифицирующие пищевые добавки на территории Российской Федерации. Рассмотрены существующие аналоги для распознавания состава продукта по фотографии. Изучены основные шаги по распознаванию текста на изображении.

Разработано клиентское приложение на Java, а также серверное приложение на Python, позволяющее распознавать состав ингредиентов и выводит информацию об их вреде для организма. Учтены случаи искаженной формы упаковки и случаи, когда часть слова не распозналась. Дальнейшая работа заключается в перенесении всех этапов обработки на клиентское приложение.

Источники

1 - ГОСТ Р 51074-2003 Продукты пищевые. Информация для потребителя. Общие требования http://ozpp.ru/standard/gosty/gostr510742003/ - проверено 01.03.2018

2 - СанПиН 2.3.2.1293-03 «Гигиенические требования по применению пищевых добавок»

3 - Combined Script and Page Orientation Estimation usingthe Tesseract OCR engine https://storage.googleapis.com/pub-tools-public-publication-data/pdf/35506.pdf - проверено 01.03.2018

4 - Taeho Kil, Wonkyo Seo, Hyung Il Koo and Nam Ik Cho, "Robust Document Image Dewarping Using Text-Line and Line Segments", ICDAR 2017 https://github.com/xellows1305/Document-Image-Dewarping - проверено 01.03.2018

5 - А. Федоров БИНАРИЗАЦИЯ ЧЕРНО-БЕЛЫХ ИЗОБРАЖЕНИЙ: СОСТОЯНИЕ И ПЕРСПЕКТИВЫ РАЗВИТИЯ http://it-claim.ru/Library/Books/ITS/wwwbook/ist4b/its4/fyodorov.htm - проверено 01.03.2018

6 - Распрямление текстовых строк на основе непрерывного гранично-скелетного представления изображений http://www.machinelearning.ru/wiki/images/e/ec/Gr_2006_Masalovitch.pdf - проверено 01.03.2018

7 - Tesseract OCR - https://github.com/tesseract-ocr/tesseract - проверено 01.03.2018

8 - Нечёткий поиск в тексте и словаре https://habr.com/post/114997/ - проверено 01.03.2018

9 - Приложение Ingred https://play.google.com/store/apps/details?id=com.zorrosoft.android.ingred&hl=ru - проверено 01.03.2018

10 - Приложение Едим - глядим https://play.google.com/store/apps/details?id=com.sergdev.omnomnom&hl=ru - проверено 01.03.2018

11 - Tesseract OCR Features and Character Classifier https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/3CharacterClassifiers.pdf - проверено 01.03.2018

12 - Tesseract OCR Character Segmentation https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/4CharSegmentation.pdf - проверено 01.06.2018

13 - Tesseract OCR Page Layout Analysis https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/5LayoutAnalysis.pdf - проверено 01.06.2018

14 - Wolf Garbe - 1000x Faster Spelling Correction algorithm https://medium.com/@wolfgarbe/1000x-faster-spelling-correction-algorithm-2012-8701fcd87a5f - проверено 01.06.2018

15 - Wojciech Bieniecki, Szymon Grabowski and Wojciech Rozenberg - Image Preprocessing for Improving OCR Accuracy, 2007

Приложение А - Отправка фото на сервер

```
        public void processPhoto() {
            ((TextView) findViewById(R.id.textView)).setText("ЗАГРУЗКА ФОТО НА
СЕРВЕР");

            new Thread(new Runnable() {
                public void run() {
                    String response;
                    try {
                        MultipartUtility multipart = new MultipartUtility("http://192.168.1.3:4444",
"UTF-8");
                        multipart.addFilePart("file", new File(mCurrentPhotoPath));
                        multipart.addFormField("bounding_box", boundingBox.flattenToString());
                        multipart.addFormField("rotation", Integer.toString(rotation));

                        response = multipart.finish();
                    } catch (Exception e) {
                        response = e.getClass().getName() + e.getMessage();
                    }

                    final String finalResponse = response;
                    runOnUiThread(new Runnable() {
                        public void run() {
                            ((TextView) findViewById(R.id.textView)).setText("");
                            renderResponse(finalResponse);
                        }
                    });
                }
            }).start();
        }
```

Приложение Б - Обработка HTTP запроса на сервере

```
from .image_parser import parse_file
from .ingredients_extractor import extract

@csrf_exempt
def do_ocr(request):
    if request.method != 'POST':
        return HttpResponseBadRequest("Not valid request")
    form = UploadFileForm(request.POST, request.FILES)
    if not form.is_valid():
        return HttpResponseBadRequest("Not valid request")
    text = parse_file(
        request.FILES['file'],
        parse_bounding_box(request.POST['bounding_box']),
        int(request.POST['rotation'])
    )
    ingredients_ids = extract(text)
    return HttpResponse(make_response_from_found_ingredients(ingredients_ids))
```

Приложение В - Обработка искажений

```
def dewarp_file(filename):
    _unused, extension = os.path.splitext(filename)
    out_filename = generate_temporary_filename(extension);
    exit_code = call([
        "wine",
        "/dewarp/Dewarping.exe",
        filename,
        out_filename,
        "0",
        "0"
    ])
    if exit_code!=0:
        raise Exception('Dewarping failed')
    return out_filename
```

Приложение Г - Распознавание текста на изображении

```
def parse_file(file, bounding_box, rotation):
    filename, file_extension = os.path.splitext(file.name)
    original_file = generate_temporary_filename(file_extension)
    save_uploaded_file(original_file, file)
    (rotated_file, rotated_box) = rotate_file(original_file, rotation, bounding_box)
    dewarped_file = dewarp_file(rotated_file)
    cropped_file = crop_dewarped_file(rotated_file, dewarped_file, rotated_box)

    found_text = pytesseract.image_to_string(cropped_file, lang='rus', config='--psm 6')
    os.remove(original_file)
    os.remove(rotated_file)
    os.remove(dewarped_file)

    return found_text
```

Приложение Д - выделение отдельных ингредиентов из текста

```
def extract(text):
    phrases = split_by_phrases(
        remove_meaningless_phrases(
            remove_newlines(
                strip_end(
                    strip_start(text)
                )
            )
        )
    )
    ingredients_ids = []
    for phrase in phrases:
        maybe_ingredient = extract_known_ingredient(phrase)
        if maybe_ingredient:
            ingredients_ids.append(maybe_ingredient)
    return ingredients_ids

def strip_start(text):
    ingredients_start_marker = u"Состав"
    index = text.find(ingredients_start_marker)
    if index==-1:
        return text
    return text[(index+len(ingredients_start_marker)):]
```

```
def strip_end(text):
    ingredients_end_markers = [
        u"Пищевая ценность",
        u"Может содержать",
        u"Продукт готов к употреблению",
        u"При чрезмерном",
        u"Способ приготовления"
    ]
    index = -1
    for marker in ingredients_end_markers:
        index = text.find(marker)
        if index != -1:
            break
    if (index==-1):
        return text
    return text[:-(len(text)-index)]
def remove_newlines(text):
    text2 = re.sub(re.compile('-*$',re.M), '', text)
    return text2.replace("\n","")
def remove_meaningless_phrases(text):
    meaningless = [
        u"регулятор кислотности",
        u"усилитель вкуса и аромата",
        u"консервант"
    ]
    for phrase in meaningless:
        text = text.replace(phrase, '')
    return text
def split_by_phrases(text):
    return filter(
        None,
        map(lambda t: t.strip(),re.split("[,:;\.\(\)]+", text))
    )
def extract_known_ingredient(phrase):
    global known_ingredients
    global symspell
    global e_pattern
    maybe_e = re.findall(e_pattern, phrase)
    if maybe_e:
        print maybe_e
        ingredient = maybe_e[0].replace(" ", "")
    else:
        ingredient = symspell.correct(phrase)
    if ingredient in known_ingredients:
        return known_ingredients[ingredient]
    else:
        return None
```