Математическое и программное обеспечение задачи оценки качества товара на основе распознавания тональности текстовых отзывов о нем.

Дипломная работа 75 с.

**Аннотация**

Дипломная работа посвящена математическому моделированию и разработке программного продукта, предназначенного для оценки качества товара основываясь на распознавании тональности текстовых отзывов из сети Интернет.

В данной дипломной работе анализ тональности текстовых отзывов осуществляется с помощью метода на основе словарей. Данный метод дополнен графематическим анализом и алгоритмом нечеткого поиска по словарю. Важной особенностью работы является реализация интеграции с интернет ресурсом Яндекс маркет.

Входными данными являются: список товаров; текстовый отзыв, описывающий товар; заданные экспертом параметры товара; словарь эмоционально окрашенных слов и словарь слов-модификаторов. Выходными данными являются класс качества товара и класс качества параметров товара.

В соответствующих разделах работы приводится описание задачи, методов ее решения, алгоритмизация выбранных методов. Также содержится описание программной документации, результаты тестирования и оценки качества созданного программного продукта. Проведены экономические расчеты цены предложения и количества копий, необходимых для окупаемости проекта.

В результате выполнения дипломной работы было разработано программное обеспечение с Web-интерфейсом, позволяющее определять класс качества товара и параметров товара.

Ил. 42, табл. 19, библ. наим. 19.

**ОГЛАВЛЕНИЕ**

ВВЕДЕНИЕ 6

ГЛАВА 1 ТЕОРЕТИКО-АНАЛИТИЧЕСКАЯ ЧАСТЬ 7

1.1. Постановка задачи 7

1.2. Анализ существующих программных средств, предназначенных для решения поставленной задачи 8

1.3. Структура решения задачи 12

1.4. Аналитический обзор моделей и методов решения 13

1.4.1. Известные подходы к решению задачи 14

1.4.2. Сравнительный анализ подходов 19

1.4.3. Выбор модели и методов решения задачи 20

1.5. Математическая постановка задачи 26

ГЛАВА 2. ПРОЕКТНАЯ ЧАСТЬ 28

2.1. Алгоритмизация методов решения задачи 28

2.2. Программная реализация 30

2.2.1. Выбор подхода к разработке программного обеспечения 30

2.2.2. Выбор языка программирования и инструментальных средств разработки 30

2.2.3. Описание структуры данных 34

2.2.4. Описание структуры программного продукта 34

2.2.5. Описание интерфейса пользователя 35

2.3. Тестирование ПО 39

2.4. Результаты экспериментальных исследований и их анализ 42

2.5. Программная документация 43

2.5.1. Техническое задание 43

2.5.2. Руководство программиста 45

2.5.3. Руководство пользователя 54

2.6. Оценка качества программного обеспечения 60

ЗАКЛЮЧЕНИЕ 74

СПИСОК ЛИТЕРАТУРЫ 75

ПРИЛОЖЕНИЕ 1. ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ 76

# ВВЕДЕНИЕ

Сегодня потребитель может приобрести любой необходимый ему товар, причем он может выбрать марку производителя и характеристики, которые для него приоритетнее. Поэтому производители находятся в условиях жесткой конкуренции. Для них важно понять, что необходимо пользователям, раньше конкурентов, или же, как улучшить свой товар. Для этого фирмы производители проводят опросы, анкетирования. Но полученная таким образом информация является труднодоступной и дорогостоящей, кроме того, она ограничивает пользователей в выражении мнений, а оцениваемые характеристики задаются заранее экспертами. В наше время с помощью глобальной сети Интернет можно узнать актуальное мнение пользователей о том или ином товаре.

То есть у фирм есть необходимость в проведении анализа потребительских характеристик товара на основе отзывов в сети Интернет. Для этого крупные фирмы создают отдел аналитиков, которые изучают отзывы о компании и товарах в Интернете. Но просмотр информации в Интернете и ее последующее преобразование затрачивает много времени и сил. Чтение объемных текстов, ручной поиск и анализ нужной информации в больших объемах текста малоэффективны. Из-за этого производители нуждаются в другом решении. Поэтому актуально разработать систему автоматической обработки отзывов в сети Интернет.

Перспективность системы автоматической обработки отзывов в сети Интернет заключается в том, что на основе текстовой информации она позволяет оценить не только отношение потребителя к определенной продукции, но и оценить успешность рекламной компании, политических и экономических реформ; выявить отношение населения к определенной персоне, организации, событию и т.п. Но, несмотря на перспективы данного направления, пока оно не столь активно. Причинами являются трудности выделения эмоциональной лексики в текстах, несовершенство существующих текстовых анализаторов, зависимость от предметной области. Поэтому совершенствование и разработка новых методов анализа тональности текста является актуальной задачей.

# ГЛАВА 1 ТЕОРЕТИКО-АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Постановка задачи

Целью дипломной работы является повышение эффективности автоматизации оценки качества товара на основе разработки соответствующего математического и программного обеспечения по распознаванию тональности текстовых отзывов из сети Интернет.

Для достижения данной цели были поставлены следующие задачи:

1. Провести обзор существующих методов автоматического анализа тональности текстовых отзывов.
2. Провести исследование особенностей текстовых сообщений в сети Интернет в контексте разработки метода анализа тональности текстовых отзывов.
3. Провести обзор и анализ алгоритмов нечеткого поиска.
4. Постановка задачи оценки качества товара в терминах теории распознавания образов.
5. Разработать метод автоматического анализа тональности текстовых отзывов из сети Интернет.
6. Разработать метод определения класса качества товара и параметров товара.
7. Разработать ПО и провести тестирование.

*Содержательная постановка задачи.*

Имеются текстовые отзывы с описанием одного товара. Текстовый отзыв о товаре можно представить как объект, обладающим рядом признаков (эмоционально окрашенные слова описывающие товар). Каждый объект необходимо отнести к определенному классу: отличный, хороший, средний, удовлетворительный, неудовлетворительный. Данную задачу можно рассматривать как задачу распознавания образов. После этого необходимо усреднить проанализированные отзывы о товаре и отнести товар к одному из классов. Аналогичную операцию проведем для каждого заданного параметра товара.

Необходимо смоделировать систему оценки качества товара основываясь на распознавании тональности текстовых отзывов из сети Интернет.

*Формальная постановка задачи.*

Формальной постановке задачи соответствует контекстная диаграмма методологии IDEF0 (Рисунок 1.1.).



Рисунок 1.1. Контекстная диаграмма.

Опишем диаграмму оценки качества товара, на основе распознавания тональности текстовых отзывов о нем.

*Имеются:*

* Текстовые отзывы.

*Механизмы исполнения:*

* Метод анализа тональности текстовых отзывов;
* Метод оценки качества товара и параметров товара.

*Управление:*

* Словарь эмоционально окрашенных слов;
* Словарь слов-модификаторов;
* Список товаров и их тезаурусы;
* Параметры товара.

*На выходе:*

* Класс качества товара;
* Класс качества параметров товара.

## Анализ существующих программных средств, предназначенных для решения поставленной задачи

Существует несколько программных продуктов, разработанных для анализа эмоциональной окраски текста. Каждый из них работает в определенной предметной области.

1. **Twitter Sentiment**

Сервис Twitter сейчас очень популярен. Пользователи сервиса, в частности, пишут в своих сообщениях отзывы о товарах. Веб-сервис Twitter Sentiment [2] позволяет анализировать информацию о продукте, который упоминают пользователи, при помощи данных из веб-сервиса Twitter.

Пользователю Twitter Sentiment достаточно ввести слово, и программа проанализирует все последние 100 записей об этом слове. При этом будет построен график соотношения положительных и негативных отзывов. В совокупности, предоставляется легкий способ проанализировать мнения пользователей о любых продуктах. Сообщения в Twitter часто содержат ошибки, сокращения тех или иных понятий и они небольшие по объему(обычно это 1 предложение).

Twitter Sentiment использует классификатор, построенный на методах машинного обучения[3], и также имеет API. Также пользователь видит, как были классифицированы сообщения. В данном сервисе реализована поддержка двух языков: английский и испанский.

В качестве примера работы сервиса использовался товар марки Samsung (Рисунок 1.2.).

Как видно из рисунка, пользователь вводит наименование продукта и выбирает язык в поле 1. После этого нажимает на кнопку «Search».

Затем в поле 2 отображаются зависимости положительных и отрицательных отзывов. На круговой диаграмме показано процентное соотношение. Рядом с ней приведено количественное соотношение.

В поле 3 показано три отзыва, которые участвовали в анализе. Причем отзыв А был классифицирован как положительный, отзыв Б – нейтральный и отзыв В – отрицательный.

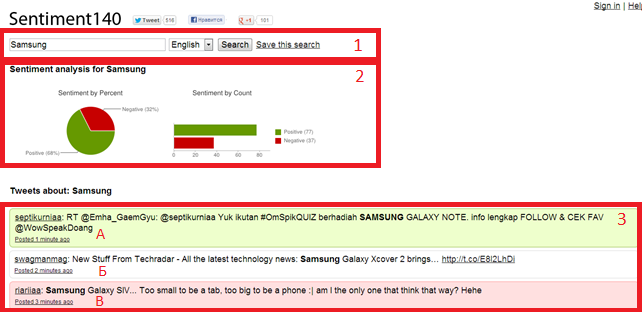


Рисунок 1.2. Пример работы сервиса Twitter Sentiment.

1. **I-Teco**

Данный программный продукт [4] является веб-сервисом. Компания «Ай-Теко» использует алгоритм анализа тональности текста, основанный на тональных словарях, состоящий из нескольких этапов: морфологического анализа текста, разметки по словарным спискам тональной лексики, синтаксического анализа и непосредственно определения тональности[1].

Данный веб-сервис позволяет анализировать текст на русском языке, содержащем не более 2000 символов. В каждом предложении выделяется объект повествования, и относительно него определяется тональность.

Сервис является универсальным, в плане предметной области, поэтому погрешность достаточно высока. Также сервис не позволяет вычислить результирующую оценку товара, хотя в тексте идет повествование об одном товаре.

Пример работы сервиса представлен на рисунке 1.3. Пользователь вводит текст в поле 1 и нажимает кнопку «Выполнить», либо если нужна корректировка «Очистить». Полученный результат отображается в поле 2. Причем предложение 1 классифицировано как положительное, предложение 2 – нейтральное, а предложение 3 – отрицательное.

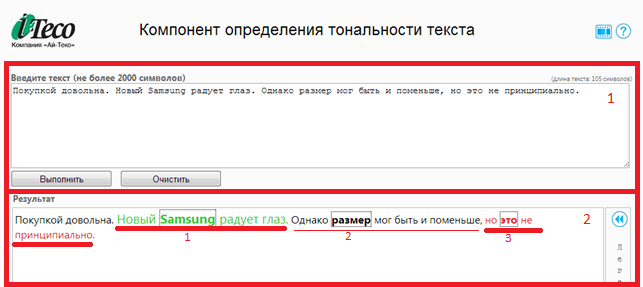


Рисунок 1.3. Пример работы сервиса I-Teco.

1. **Sentiment Analysis with Python NLTK Text Classification**

Данный программный продукт имеет веб-интерфейс [5]. Библиотека NLTK, или NTLK (Natural Language Toolkit) — пакет библиотек и программ для символьной и статистической обработки естественного языка, написанных на языке программирования Python.

Веб-сервис используется классификатор, построенный на методе машинного обучения[3] с применением наивного байесового классификатора[6]. Веб-сервис поддерживает 2 языка: английский и немецкий. Текст может содержать максимум 50000 символов. Также данный сервис предназначен для классификации отзывов о фильмах. В других предметных областях сервис работает с погрешностью.

Пример работы сервиса представлен на рисунке 1.4. Также как и в предыдущем веб-сервисе пользователю необходимо ввести текст в поле 1, выбрать язык и нажать на кнопку «Analyze». Результаты анализа показаны в окне 2. Причем финальная классификация основана на вероятностях субъективности (нейтральная или полярная) и полярности (негативная или отрицательная).

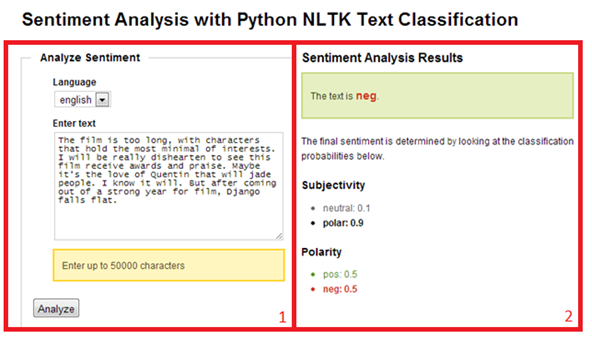


Рисунок 1.4. Пример работы сервиса Sentiment Analysis with Python NLTK Text Classification.

Мною были проанализированы некоторые из программных продуктов, которые могут использоваться для поставленной задачи. В таблице 1.1. представлены основные характеристики описанных выше программных продуктов.

Таблица 1.1.Анализ программных средств, предназначенных для решения поставленной задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Программные  продукты  Характеристики | | Twitter Sentiment | I-Teco | Sentiment Analysis with Python NLTK Text Classification |
| Пользовательские характеристики | Язык для анализа | Английский, испанский | Русский | Английский, немецкий |
| Ограничения по объему текста | Сообщение в социальной сети Twitter | 2000 символов | 50000 символов |
| Стоимость | 200$ - коммерческая лицензия | 0$ | Demo – 0$. |
| Финальная классификация | + | - | + |
| Технические характеристики | Язык программирования | JavaScript | Java | Python |
| Метод анализа тональности текста | Метод машинного обучения без учителя | Подход на основе словарей | Метод машинного обучения с учителем |
| Точность анализа | 65% | 70% | 70% |

Изучив имеющиеся программные продукты, можно сделать следующие вывод: ни один из представленных программных продуктов не решает поставленной задачи в целом. Во-первых, русский язык рассматривается только в одном программном продукте. Во-вторых, рассмотренные ПО не являются универсальными и для конкретной предметной области не подходят из-за отсутствия нужных словарей. В-третьих, в представленных выше программных продуктах отсутствует промежуточная и финальная классификация, то есть пользователь не может проследить весь цикл вычисления оценки конкретного товара. В-четвертых, у данных ПО отсутствует выделение заданных параметров товара и их оценка качества.

Поэтому мною было решение разработать собственное программное обеспечение, которое бы решало поставленные выше задачи.

## Структура решения задачи

Перед тем, как начать разрабатывать математическую модель и программное обеспечение, необходимо провести подготовительную работу.

1. Выберем:
   1. предметную область;
   2. язык, на котором будет проводиться анализ.
2. Составим словари (таблицы в базе данных):
   1. эмоционально окрашенных слов;
   2. Слов-модификаторов;
   3. товаров, характеризующих выбранную предметную область;
      1. тезаурусы для каждого товара;
   4. параметров товаров.
3. Проведем исследование текстовых особенностей сообщений в сети Интернет в контексте разработки методов анализа их тональности.
4. Проведем обзор и анализ алгоритмов нечеткого поиска.
5. Проведем обзор существующих методов автоматического анализа тональности текстов.

После проведения подготовительной работы приступим к:

1. разработке метода анализа тональности текстовых отзывов из сети Интернет;
2. разработке метода оценки качества товара и параметров товара;
3. разработке программного продукта, реализующего метод анализа тональности текстовых отзывов и метод оценки товара и его параметров;

Структура решения задачи раскрывается на диаграмме декомпозиции, построенная согласно методологии IDEF0 (Рисунок 1.5.).



Рисунок 1.5.Структура решения задачи.

## Аналитический обзор моделей и методов решения

Ключевое значение в дипломном проектировании является обработка текстовых отзывов с целью выделения в них упоминания товаров и их характеристик. Относительно этого будем рассматривать известные методы к решению задачи анализа тональности текста.

Sentiment analysis, или анализ тональности текста, – это развивающееся направление компьютерной лингвистики, основной задачей которого является выявление в документе эмоционально окрашенной лексики и эмоциональной оценки объектов автором. Эмоциональная оценка, выраженная в тексте, также называется тональностью, или сентиментом текста. Лексическая тональность (или лексический сентимент) – эмоциональная составляющая, выраженная на уровне лексемы. Эмоциональная окраска текста определяется тональностью его составляющих, а также их взаимосвязями [1].

### Известные подходы к решению задачи

Для автоматического определения тональности текста можно выделить следующие методы:

1. Метод на основе правил.
2. Метод на основе словарей.
3. Машинное обучение с учителем.
4. Машинное обучение без учителя.
5. Гибридный метод.

Рассмотрим каждый подход подробнее.

*Метод на основе правил (rule-based with patterns).*

Данный метод заключается в генерации правил, на основе которых будет определяться тональность текста [7]. Для этого текст разбивается на слова или комбинации слов (N-граммы) и, затем, полученные данные используются для выделения встречающихся шаблонов. Им присваиваются положительная или отрицательная оценка. Выделенные шаблоны применяются при создании правил вида: ЕСЛИ «условие», ТО «заключение».

Пример.

В тексте встретилось предложение «Я люблю кофе». Можно применить следующее правило:

**ЕСЛИ** *сказуемое («люблю») входит в положительный набор глаголов («люблю», «обожаю», «одобряю» ...)* **И** *в предложении не имеется отрицаний*,

**ТО** *классифицировать тональность как «положительная».*

*Метод на основе словарей (affective lexicons).*

Метод для анализа текста основан на тональных словарях(affective lexicons). В простом виде словарь представляет список слов со значением тональности для каждого слова. Тональные словари состоят из таких элементов, как слова, словосочетания, каждые из которых имеют свою эмоциональную окраску. Тональность текста определяется по совокупности найденной эмоциональной лексики и оценивается в зависимости от количества позитивных и негативных слов.

*Машинное обучение с учителем (supervised learning).*

Данный метод является наиболее распространенным. Его суть заключена в том, чтобы обучить машинный классификатор на коллекции заранее размеченных текстах. И затем использовать полученную модель для анализа новых документов [8].

*Машинное обучение без учителя (unsupervised learning).*

В основе данного метода лежит идея, что наибольший вес в тексте имеют термины, которые чаще всего встречаются в тексте. Выделив эти термины и определив их тональность можно сделать выводы обо всем тексте [9].

*Гибридный метод (hybrid method).*

Данный метод включает в себе выше перечисленные принципы и заключается в применении классификаторов на их основе [10].

Рассмотрим алгоритмы нечеткого поиска в тексте и словаре.

Алгоритмы нечеткого поиска *(fuzzy string search)* являются основой систем проверки орфографии и полноценных поисковых систем Google, Yandex.

Нечеткий поиск является крайне полезной функцией любой поисковой системы. Вместе с тем, его эффективная реализация намного сложнее, чем реализация простого поиска по точному совпадению.

Задачу нечеткого поиска можно сформулировать так:

«По заданному слову найти в тексте или словаре размера n все слова, совпадающие с этим словом (или начинающиеся с этого слова) с учетом k возможных различий».

Например, при запросе «Машина» с учетом двух возможных ошибок, найти слова «Машинка», «Махина», «Малина», «Калина» и так далее.

Алгоритмы нечеткого поиска характеризуются метрикой — функцией расстояния между двумя словами, позволяющей оценить степень их сходства в данном контексте. Строгое математическое определение метрики включает в себя необходимость соответствия условию неравенства треугольника (X — множество слов, p — метрика):

Между тем, в большинстве случаев под метрикой подразумевается более общее понятие, не требующее выполнения такого условия, это понятие можно также назвать расстоянием.

В числе наиболее известных метрик — расстояния Хемминга, Левенштейна и Дамерау-Левенштейна.

1. *Расстояние Хемминга*

Расстояние Хэмминга[11] — число позиций, в которых соответствующие символы двух слов одинаковой длины различны. В более общем случае расстояние Хэмминга применяется для строк одинаковой длины любых q-ичных алфавитов и служит метрикой различия (функцией, определяющей расстояние в метрическом пространстве) объектов одинаковой размерности.

Расстояние Хэмминга является частным случаем метрики Минковского (при соответствующем определении вычитания). Вычисляется по формуле (\*).

(\*)

1. *Расстояние Левенштейна*

Наиболее часто применяемой метрикой является расстояние Левенштейна[12], или расстояние редактирования.

Наиболее популярного алгоритма расчета — метода Вагнера-Фишера[14].

Исходный вариант этого алгоритма имеет временную сложность **O(mn)** и потребляет **O(mn)** памяти, где m и n — длины сравниваемых строк. Весь процесс можно представить следующей матрицей(рисунок1.6.):

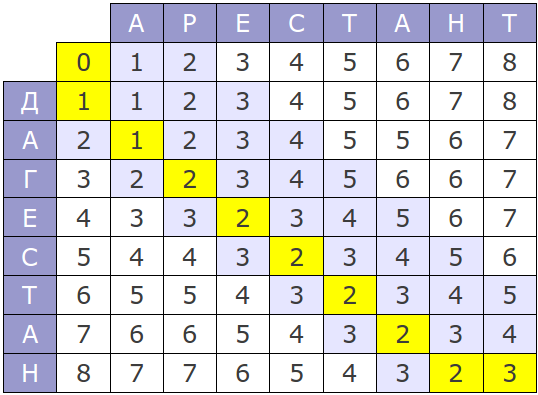


Рисунок 1.6. Процесс работы метода Вагнера-Фишера.

Если рассмотреть процесс работы алгоритма, несложно заметить, что на каждом шаге используются только две последние строки матрицы, следовательно, потребление памяти можно уменьшить до **O(min(m, n))**. Расчет расстоянияЛевенштейна производится по формуле (\*\*). Пример фрагмента матрицы на рисунке 1.7.

(\*\*)

*где*

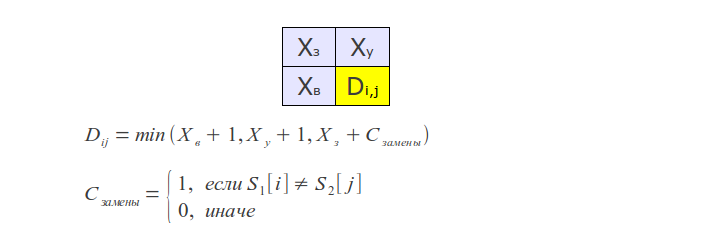


Рисунок 1.7. Фрагмент матрицы расчета расстояния Левенштейна.

Если стоит задача нахождения не более k различий. В этом случае нужно вычислять в матрице лишь диагональную полосу шириной 2k+1 (отсечение Укконена), что сводит временную сложность к **O(k min(m, n))**.

1. *Расстояние Дамерау-Левенштейна*

Расстояние Дамерау-Левенштейна[13] – это вариация вносит в определение расстояния Левенштейна еще одно правило — транспозиция (перестановка) двух соседних букв также учитывается как одна операция, наряду со вставками, удалениями и заменами.

Расчет расстоянияДамерау-Левенштейна производится по формуле (\*\*\*). Пример фрагмента матрицы на рисунке 1.8.

(\*\*\*)

*где*

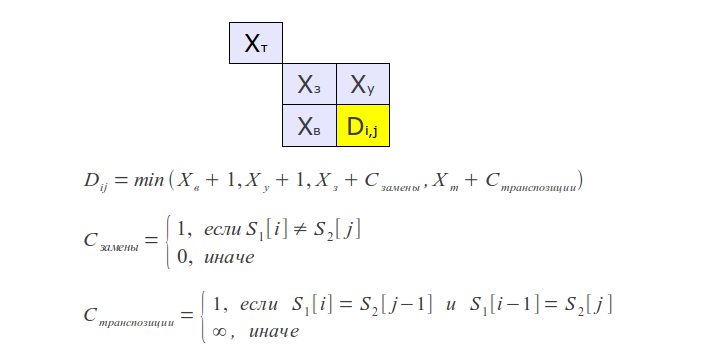


Рисунок 1.8. Фрагмент матрицы расчета расстояния Дамерау-Левенштейна.

Чтобы вычислять такое расстояние, достаточно немного модифицировать алгоритм нахождения обычного расстояния Левенштейна следующим образом: хранить не две, а три последних строки матрицы, а также добавить соответствующее дополнительное условие — в случае обнаружения транспозиции при расчете расстояния также учитывать и её стоимость.

Рассмотрим основные алгоритмы нечеткого поиска.

1. *Алгоритм расширения выборки.*

Этот алгоритм часто применяется в системах проверки орфографии (spell-checker), там, где размер словаря невелик, либо же где скорость работы не является основным критерием. Он основан на сведении задачи о нечетком поиске к задаче о точном поиске. Из исходного запроса строится множество «ошибочных» слов, для каждого из которых затем производится точный поиск в словаре.

Время его работы сильно зависит от числа *k* ошибок и от размера алфавита *A*, и в случае использования бинарного поиска по словарю составляет:

Например, при *k = 1* и слова длины 7 (например, «Крокодил») в русском алфавите множество ошибочных слов будет размером около 450(рисунок 1.9.), то есть будет необходимо сделать 450 запросов к словарю, что вполне приемлемо.

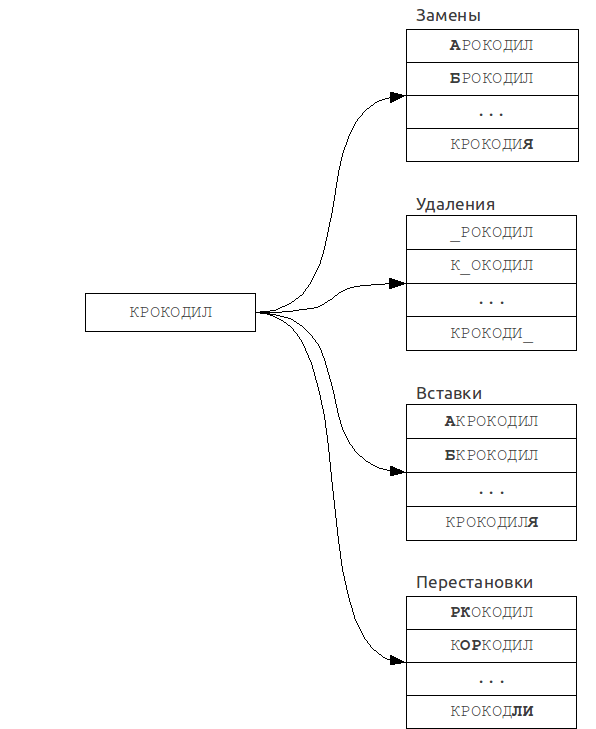


Рисунок 1.9. Пример работы алгоритма расширения выборки.

*Особенности:*

Алгоритм может быть легко модифицирован для генерации «ошибочных» вариантов по произвольным правилам, и, к тому же, не требует никакой предварительной обработки словаря, и, соответственно, дополнительной памяти.

*Возможные улучшения:*

Можно генерировать не всё множество «ошибочных» слов, а только те из них, которые наиболее вероятно могут встретиться в реальной ситуации, например, слова с учетом распространенных орфографических ошибок или ошибок набора.

1. *Метод N-грамм.*

Этот метод является наиболее широко используемым, так как его реализация крайне проста, и он обеспечивает достаточно хорошую производительность[15]. Алгоритм основывается на принципе: *«Если слово А совпадает со словом Б с учетом нескольких ошибок, то с большой долей вероятности у них будет хотя бы одна общая подстрока длины N».*

Эти подстроки длины N и называются N-граммами.

Во время индексации слово разбивается на такие N-граммы, а затем слово попадает в списки для каждой из этих N-грамм. Во время поиска запрос также разбивается на N-граммы, и для каждой из них производится последовательный перебор списка слов, содержащих такую подстроку. Пример работы алгоритма показан на рисунке 1.10.

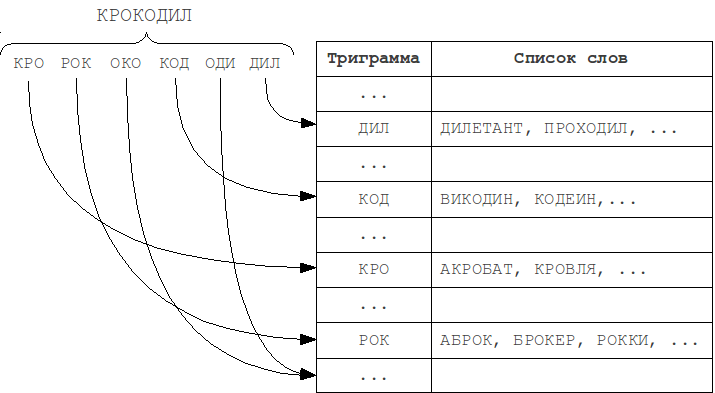


Рисунок 1.10. Пример работы метода N-грамм.

Наиболее часто используемыми на практике являются триграммы — подстроки длины 3. Выбор большего значения N ведет к ограничению на минимальную длину слова, при которой уже возможно обнаружение ошибок.

*Особенности:*

Алгоритм N-грамм находит не все возможные слова с ошибками. Если взять, например, слово ВОТКА, и разложить его на триграммы: ВОТКА → ВОТ ОТК ТКА — то можно заметить, что они все содержат ошибку Т. Таким образом, слово «ВОДКА» найдено не будет, так как оно не содержит ни одной из этих триграмм, и не попадет в соответствующие им списки. Таким образом, чем меньше длина слова и чем больше в нем ошибок, тем выше шанс того, что оно не попадет в соответствующие N-граммам запроса списки, и не будет присутствовать в результате.

*Возможные улучшения:*

Можно разбивать хеш-таблицы N-грамм по длине слов и по позиции N-граммы в слове. Как длина искомого слова и запроса не могут отличаться более чем на *k*, так и позиции N-граммы в слове могут различаться не более чем на *k*. Таким образом, необходимо будет проверить лишь таблицу, соответствующую позиции этой N-граммы в слове, а также *k* таблиц слева и *k* таблиц справа, т.е. всего *2k+1* соседних таблиц.

### Сравнительный анализ подходов

Рассмотрим преимущества и недостатки приведенных выше методов к определению тональности текста.

Многие коммерческие системы используют метод на основе правил. Однако, он требует больших затрат, т.к. для хорошей работы системы необходимо составить большое количество правил, которые чаще всего относятся к определенной предметной области и, при ее смене, требуется заново составлять правила. Тем не менее, этот метод является наиболее точным при наличии хорошей базы правил.

Другой метод на основе словарей также является трудоемким и требует составления тональных словарей, получения списка тональных паттернов и разработки лингвистических анализаторов. Но он отличается от предыдущего большей гибкостью, при этом давая результаты с большой точностью.

Метод, основанный на использовании машинного обучения с учителем, предполагает наличие предварительно размеченного обучающего набора данных. Целью обучения в случае анализа тональности является получение необходимых и достаточных правил, с помощью которых можно произвести классификацию по тональности новых текстовых сообщений, сходных с теми, которые составляли обучающую выборку. Недостатком алгоритмов анализа тональности на основе машинного обучения является зависимость от качества и количества обучающих данных, к тому же метод не позволяет провести глубокий анализ текста, выявить объект и субъект тональности.

Метод обучения без учителя не требует данные для обучения и является автоматическим. Однако в данном методе есть значительный недостаток: словам, встречающимся чаще всего в исследуемом тексте, ставится в соответствие та или иная тональность. Данный метод хорош для автоматической кластеризации документов. Но для определения тональности текста он наименее точен.

Гибридный метод является весьма перспективным направлением. Однако для каждой предметной области необходимо выбирать методы, которые будут совмещаться в данном методе. И заранее не известно, какие результаты будут получены, то есть необходимо проводить множество тестов.

Выберем алгоритм расширения выборки для реализации нечеткого поиска. Т.к. данный метод дает более точные результаты, чем другие методы.

При выборе метрики, используемой в алгоритме нечеткого поиска, наиболее полная – Дамерау-Левенштейна. Так как большинство ошибок при наборе текста — транспозиции. Поэтому именно данная метрика дает наилучшие результаты на практике.

### Выбор модели и методов решения задачи

В данной дипломной работе разработан модифицированный метод на основе составления словарей, дополненный графематическим анализом и алгоритмом нечеткого поиска по словарю.

Для уменьшения текстовых отзывов, была выбрана предметная область – мобильные телефоны и язык анализа – русский.

***Формирование тонального словаря.***

Для решения задачи автоматического анализа тональности текста был создан *словарь* *эмоционально окрашенных слов*, характерной для данной предметной области. Изначально вручную было отобрано 50 оценочных слов и поставлены им веса по шкале [-5;0) для отрицательно ориентированных слов и [0;5] для положительно ориентированных слов. Далее использовался метод, использующий тезаурус, для пополнения заданного вручную множества оценочных слов. Идея метода заключена в том, что если у оценочного слова есть синонимы, то они также будут внесены в словарь и иметь такую же тональность. В случае если у оценочного слова есть антонимы, то они тоже будут занесены в словарь с противоположной тональностью. Также данный словарь был заполнен словосочетаниями и жаргонизмами, наиболее часто употребляемыми для описания выбранной предметной области.

Для универсального хранения эмоционально окрашенных слов была использована библиотека, реализующая алгоритм стэмминга, Snowball stemmer[16].

Кроме этого необходимо создать *словарь слов-модификаторов*. Это слова, в зависимости от которых увеличивается или уменьшается вес следующего за ним эмоционально окрашенного слова. То есть все слова-модификаторы можно разделить на две группы. К первой группе относятся слова, увеличивающие вес соседнего слова (например *более*), а ко второй – слова, уменьшающие вес (например *менее*).

Для изменения веса слова следующего за словом-модификатором будем использовать подход, который в зависимости от слова-модификатора изменяет вес соседнего слова на некоторый процент. Например, если слово *функционально* имеет вес 2, то *более функционально* будет иметь вес .

А также необходимо учитывать частицы и слова, используемые в тексте для отрицания. Очевидным подходом к учету слов, выражающих отрицание, является инвертирование веса эмоционально окрашенного слова, находящегося после слова, выражающего отрицание. Например, слово *функциональный* имеет вес 2, а значит *не* *функциональный* будет иметь вес -2. В качестве слов, выражающих отрицание, используются частицы *не*, *ни*, и местоимение *ничего*.

***Подготовка списка товаров и их параметров.***

Определим список товаров для их анализа. Для них всех определим список заданных параметров, которые наиболее важны в оценке данных товаров. Для предметной области мобильные телефоны, эти параметры перечислены в таблице 1.2.

Таблица 1.2.Отобранные параметры товаров

|  |  |
| --- | --- |
| Внешний вид: | Внутреннее наполнение: |
| корпус;  дисплей;  сборка;  вес;  габариты. | операционная система;  батарея;  процессор;  динамики;  микрофон;  камера;  сенсор;  связь;  звук. |

После того как были составлены словари, список товаров и их параметров, можно приступать к определению тональности текста.

***Метод автоматического анализа тональности текстовых отзывов.***

В данной дипломной работе тональность текстовых отзывов определяется на основе подсчета весов входящих в него эмоционально окрашенных слов. Веса эмоционально окрашенных слов извлекаются из словаря, формирование которого было описано выше.

*Нечеткий поиск в тексте и словаре.*

При поиске эмоционально окрашенных слов будем использовать Snowball stemmer, что позволит нам убрать зависимость от формы слова. Для исключения орфографических ошибок в эмоционально окрашенных словах используется алгоритм расширения выборки с расстоянием Дамерау-Левенштейна.

Алгоритм основан на сведении задачи о нечетком поиске к задаче о точном поиске. Из исходного запроса строится множество «ошибочных» слов, для каждого из которых затем производится точный поиск в словаре.

Время его работы сильно зависит от числа ошибок *k* и от размера алфавита *A*, и в случае использования бинарного поиска по словарю составляет:

Пусть количество ошибок в слове *k=1*, размер алфавита русского языка *A=33*.

Расчет расстоянияДамерау-Левенштейна производится по формуле (1), где – число вставок, – число удалений, – число замен, – число транспозиций. Пример фрагмента матрицы на рисунке 1.11.

(1)

*где*

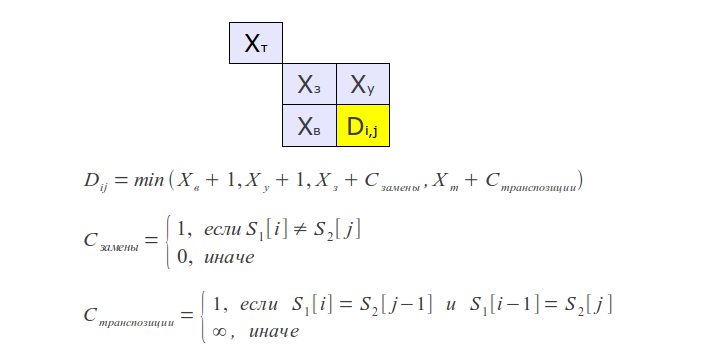


Рисунок 1.11. Фрагмент матрицы расчета расстояния Дамерау-Левенштейна.

Пример работы алгоритма нечеткого поиска для слова «Хороший» показан на рисунке 1.12.

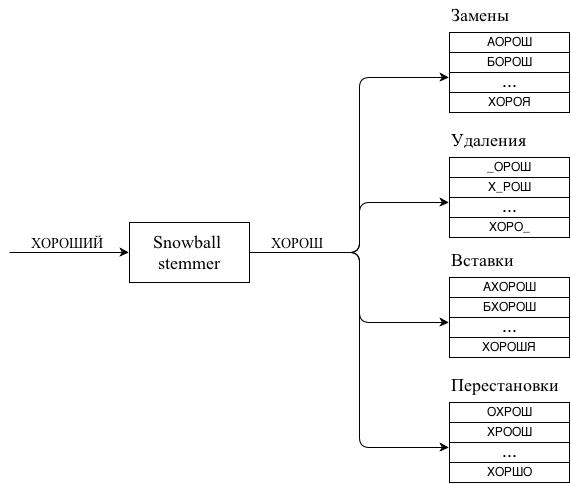


Рисунок 1.12. Пример работы алгоритма нечеткого поиска.

После генерации возможных вариантов слов, осуществляется поиск по словарям в базе данных.

*Расчет результирующей оценки отзыва.*

После того, как были выявлены эмоционально окрашенные слова, входящие в отзыв. Для каждого текстового отзыва подсчитывается его оценка, равная среднему весу входящих в него оценочных слов:

(2)

где – оценка текстового отзыва, – вес оценочного слова *i*; *N* – количество оценочных слов в отзыве *T*.

Слова-модификаторы и слова, выражающие отрицание, не являются оценочными словами, а изменяют вес оценочных слов при вычислении оценки отзыва.

Рассмотрим следующие конструкции отзывов:

1. Функциональный интерфейс.

*Оценка предложения равна весу слова функциональный, то есть* ***2****.*

1. Очень функциональный интерфейс.

*Оценка предложения равна весу слова функциональный, увеличенному на 50% из-за слова-модификатора очень, то есть* ***2\*(1+0,5) = 3****.*

1. Не функциональный интерфейс.

*Оценка предложения равна весу слова функциональный, инвертированному из-за частицы не, то есть -****2****.*

1. Не очень функциональный интерфейс.

*Оценка предложения равна весу слова функциональный, уменьшенному на 50% из-за слова-модификатора очень и инвертированному из-за частицы не, то есть* ***-2\*(1-(0,5)) = -1****.*

1. Очень не функциональный интерфейс.

*Оценка предложения равна весу слова функциональный, уменьшенному на 25%(50% делим на 2) из-за слова-модификатора очень и инвертированному из-за частицы не, то есть* ***-2\*(1-(0,5/2)) = -1,5****.*

Если не было выделено ни одной из представленных выше конструкций, то система не может классифицировать отзыв.

Далее необходимо определить к какому классу относится проанализированный отзыв. Для этого сопоставим рассчитанную оценку отзыва с диапазонами оценок классов. Выберем тот класс, в который попадает оценка отзыва. Зададим следующее разбиение на классы *W*(таблица 1.3):

Таблица 1.3.Разбиение на классы

|  |  |  |
| --- | --- | --- |
| Обозначение класса | Название класса | Значение оценок, соответствующих классу |
| *W1* | «Отличный» | [5;3] |
| *W2* | «Хороший» | (3;1] |
| *W3* | «Средний» | (1;-1) |
| *W4* | «Удовлетворительный» | [-1;-3) |
| *W5* | «Неудовлетворительный» | [-3;-5] |

*Расчет оценки параметра товара в отзыве.*

В отзыве часто фигурируют определенные параметры, которые наиболее важны для выбранной предметной области. Поэтому важно, что бы система выделяла упоминание этих параметров в тексте и искала эмоционально окрашенные слова, связанные с ними. Для данной цели используется open-source library FreeLing[17]. Данная библиотека помогает выполнять графематический анализ текста с целью выявления в тексте зависимостей между словами. Для каждого предложения стоится граф.

Рассмотрим следующее предложение: *У телефона отличный дизайн и быстрая ОС*.(Рисунок 1.13.).

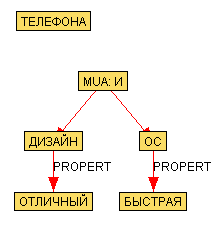


Рисунок 1.13.Граф предложения, построенный с помощью open-source library FreeLing.

Затем выделяются, заранее заданные параметры товара, и для них вычисляется оценка по формуле 3.

(3)

где – оценка параметра товара, – вес оценочного слова, относящегося к параметру в текстовом отзыве *i*; *N* – количество оценочных слов, относящихся к параметру товара *P*.

То есть мы рассматриваем часть текстового отзыва, содержащую описание параметра товара(задача аналогичная вышеописанной). Далее необходимо определить к какому классу относится часть текстового отзыва, содержащая описание параметра товара. Для этого сопоставим рассчитанную оценку части отзыва с диапазонами оценок классов. Выберем тот класс, в который попадает оценка части отзыва. Будем использовать разбиение на классы *W* описанные в таблице 1.3:

***Метод оценки качества товара и параметров товара.***

*Определение оценки качества товара.*

Для каждого товара подсчитывается его оценка, равная средней оценке описывающих его текстовых отзывов:

(4)

где *R* – оценка товара, – оценка текстового отзыва *T*; *M* – количество текстовых отзывов о товаре *L*.

*Определение оценки качества параметров товара.*

Качество параметров товара производится аналогично. Для каждого параметра будем считать оценку по формуле 5.

(5)

где *R* – оценка параметра товара, – оценка параметра товара в текстовом отзыве *T*; *M* – количество текстовых отзывов с упоминанием параметра товара *P*.

Качество товара и его параметров определяется путем выбора класса, к которому относится его оценка. Будем использовать разбиение на классы *W* описанные в таблице 1.3:

## Математическая постановка задачи

Математическая постановка задачи анализа тональности текстовых отзывов в терминах теории распознавания образов.

*Задача классификации.*

Пусть имеется множество текстовых отзывов – множество объектов распознавания. Имеется непересекающиеся подмножества – классы . Соотношение между объектами и классами задается индикаторной функцией . Также имеется множество эмоционально окрашенных слов – пространство признаков, воспринимаемых наблюдателем.

– функция, ставящая в соответствие каждому объекту *ti* точку *x(t)* в пространстве признаков. Векторы – образы объектов, воспринимаемые наблюдателем.

Имеется – решающее правило – оценка для на основании , т.е. .

**Дано:** текстовые отзывы , 5 непересекающихся классов , эмоционально окрашенные слова .

**Требуется:** определить класс *W* каждого объекта из множества *T*.

Математическая постановка задачи нечеткого поиска в словаре в терминах теории распознавания образов.

*Задача нечеткого поиска.*

Пусть имеется алфавит *A = {a1 ,a2 ,a3 ,…as }*, *|A| = s, s>1* – любое конечное множество символов. Слово *w* в алфавите *A* длины *n* определяется как конечная последовательность символов алфавита *A*. *w=<w1 w2 w3 …wn >*, *wi A, i=1,...,n*. Словарем именных компонент *U* будем называть заданное конечное множество слов, определённых на алфавите *A*: *U = {u1 ,u2 ,u3 ,…um }*, *|U|=m, m1*. Пусть *k* – число допустимых ошибок в слове.

С учетом этого, рассматриваемая в работе задача может быть формально поставлена следующим образом.

**Дано:** слово-образец *w*, словарь *U*, целое *k > 0* и функция расстояния *d*, являющаяся метрикой.

**Требуется:** найти все *j* такие, что *d(w, uj) k*, *ujU*, где *d* – расстояние Дамерау-Левенштейна, определяемое как минимальное количество операций замены, вставки, удаления и транспозиции символов, необходимых для преобразования одного слова в другое.

# ГЛАВА 2. ПРОЕКТНАЯ ЧАСТЬ

## Алгоритмизация методов решения задачи

Для рассмотренных выше этапов решения задачи этап алгоритмизации необходим в задаче автоматического анализа тональности текстовых отзывов и определения оценки качества товара и параметров товара.

На рисунке 2.1. приведен укрупненный алгоритм анализа текстовых отзывов и расчета оценки товара и оценок параметров товара.

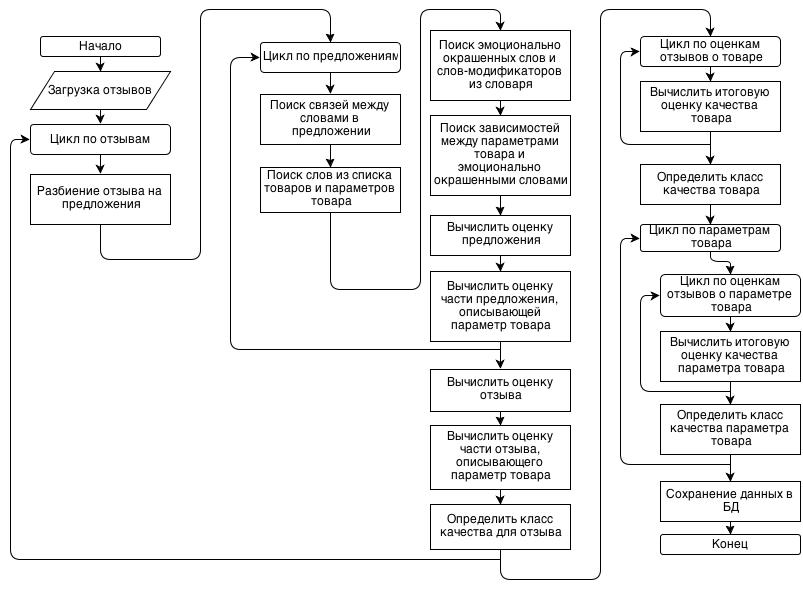


Рисунок 2.1. Блок схема укрупненного алгоритма анализа текстовых отзывов и расчета оценки качества товара и параметров товара.

*Детализированный алгоритм.*

1. Загрузка последних отзывов о выбранных товарах.
2. В цикле по отзывам:
   1. Разбить отзыв на предложения.
   2. В цикле по предложениям:
      1. Поиск связей между словами в предложении с помощью графематического анализа.
      2. Поиск слов, описывающих товар и параметры товара, из имеющегося списка в БД.
      3. Поиск эмоционально окрашенных слов и слов-модификаторов.
      4. Поиск зависимостей между параметрами товара и эмоционально окрашенными словами.
      5. В цикле для каждого зависимого слова из пункта 2.2.4:
         1. Стеммировать слово.
         2. Вычислить расстояние Дамерау-Левенштейна.
         3. Составить возможные вариации слова.
         4. Найти в БД вес зависимого слова.
      6. Для каждой тройки (товар–эмоционально окрашенное слово–слово-модификатор):
         1. Вычислить оценку качества товара.
         2. Запомнить оценку качества товара.
      7. Для каждой тройки (параметр товара–эмоционально окрашенное слово–слово-модификатор):
         1. Вычислить оценку качества параметра товара.
         2. Запомнить оценку качества параметра товара.
      8. Вычислить оценку качества товара в предложении.
      9. Вычислить оценку качества параметра товара в предложении.
   3. Вычислить оценку качества товара в отзыве.
   4. Вычислить оценку качества параметра товара в отзыве.
   5. Сохранить данные в БД.
3. В цикле по оценкам отзывов о товаре:
   * 1. Вычислить итоговую оценку качества товара.
     2. Определить класс качества товара.
     3. Сохранить данные в БД.
4. В цикле по параметрам товара:
   * 1. В цикле по оценкам отзывов о параметре товара:
        1. Вычислить итоговую оценку качества параметра товара.
        2. Определить класс качества параметра товара.
        3. Сохранить данные в БД.

## Программная реализация

### Выбор подхода к разработке программного обеспечения

Существует два основных подхода к разработке ПО: процедурный и объектно-ориентированный. Каждый из них имеет свои преимущества и недостатки. При использовании процедурного подхода, обычно появляется огромное количество глобальных переменных, и структурирование происходит на уровне модулей (файлов). В объектно-ориентированном подходе структурирование происходит на уровне объектов классов с использованием таких возможностей как шаблоны программирования, например интерфейс, абстракция и т. д.

Большинство современных языков высокого уровня поддерживают оба этих подхода.

В данной работе был использован как объектный, так и процедурный подход, т.к. большинство функциональных элементов являются унаследованными от стандартных классов.

### Выбор языка программирования и инструментальных средств разработки

* + - 1. **Характер решаемой задачи**

Решаемая задача должна быть построена с использованием технологий, для использовании ее в сети Интернет. Основная ее задача оценка качества товара на основе распознавании тональности текстовых отзывов из сети Интернет.

* + - 1. **Выбор СУБД**

При выборе базы данных необходимо выбрать базу данных, которая в наибольшей степени соответствуют предъявляемым к информационной системе требованиям. Приведем достоинства и недостатки подходов к организации хранения информации в базе данных.

**Иерархическая модель.**

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией.

Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

**Сетевая модель.**

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей.

Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

**Реляционная модель.**

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы же эффективности обработки данных этого типа оказались технически вполне разрешимыми.

Основными недостатками реляционной модели являются следующие: отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

**Постреляционная модель.**

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостатком постреляционной модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных.

**Многомерная модель.**

Основным достоинством многомерной модели данных является удобство и эффективность аналитической обработки больших объемов данных, связанных со временем. При организации обработки аналогичных данных на основе реляционной модели происходит нелинейный рост трудоемкости операций в зависимости от размерности БД и существенное увеличение затрат оперативной памяти на индексацию.

Недостатком многомерной модели данных является ее громоздкость для простейших задач обычной оперативной обработки информации.

**Объектно-ориентированная модель.**

Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных связях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки.

Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

В решаемой предметной области отсутствуют сложные иерархические связи между выделенными сущностями. Учитывая достоинство реляционной модели данных, заключенная в простоте и понятности физической реализации на ЭВМ, она была выбрана основной моделью для решения задачи.

В качестве хранения данных был выбран MySQL SERVER 5.5.28.

* + - 1. **Выбор языка программирования**

**Delphi**

Delphi — язык программирования, который используется в одноимённой среде разработки. Сначала язык назывался Object Pascal. Начиная со среды разработки Delphi 7.0, в официальных документах Borland стала использовать название Delphi для обозначения языка Object Pascal.

Для подключения к СУБД Oracle используется технология ADO.

**C++ Builder**

C++Builder – программный продукт, инструмент быстрой разработки приложений (RAD), интегрированная среда программирования (IDE), система, используемая программистами для разработки программного обеспечения на языке C++. C++ Builder объединяет в себе комплекс объектных библиотек (STL, VCL, CLX, MFC и др.), компилятор, отладчик, редактор кода и многие другие компоненты. Цикл разработки аналогичен Delphi. Большинство компонентов, разработанных в Delphi, можно использовать и в C++ Builder без модификации, но, к сожалению, обратное утверждение не верно.

C++ Builder содержит инструменты, которые при помощи drag-and-drop действительно делают разработку визуальной, упрощает программирование благодаря встроенному WYSIWYG - редактору интерфейса и пр.

Для подключения к СУБД Oracle используется технология ADO.

**C#**

C# — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как основной язык разработки приложений для платформы Microsoft .NET. Компилятор с C# входит в стандартную установку самой .NET, поэтому программы на нём можно создавать и компилировать даже без инструментальных средств, вроде Visual Studio.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Java, Delphi, Модула и Smalltalk — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++).

Для доступа к базам данных используется технология ADO.NET. ADO.NET (ActiveX Data Objects .NET) является набором классов, реализующих программные интерфейсы для облегчения подключения к базам данных из приложения независимо от особенностей реализации конкретной системы управления базами данных и от структуры самой базы данных, а также независимо от места расположения этой самой базы — в частности, в распределенной среде (клиент-серверное приложение) на стороне сервера.

**Java**

Концепция языка Java, как совокупности языка программирования и виртуальной машины, ведет свое происхождение от проекта фирмы Sun под названием Green (Oak), открытого в 1990 г. Первоначальной целью проекта было создать среду разработки программного обеспечения для бытовой электроники. Компания Sun поставила задачу заменить множество разнообразных архитектур микроконтроллеров одной единственной масштабируемой архитектурой (прототипом виртуальной Java-машины). Затем проект был переориентирован на World Wide Web и в августе 1995 г. состоялся первый официальный выпуск Java. Своим стремительным распространением в широких кругах сообщества Java обязана необычайно удачному сочетанию следующих факторов: быстрого роста сети Internet и телекоммуникационных технологий, возрастанию потребностей образования в новых подходах к программированию, успехам в создании мобильных, портативных вычисли- тельных устройств.

Принципиальные нововведения, которые отличают Java от других: независимость небольших мобильных программ в сочетании с генерацией кода в процессе выполнения, переносимость, строгая типизация с поддержкой динамических типов и системы «сборки мусора». Java предоставляет мощные объектно-ориентированные принципы разработки приложений, сочетая простой и ясный синтаксис с надежной и удобной в работе системой программирования, что позволяет быстро обучаться и создавать новые программы.

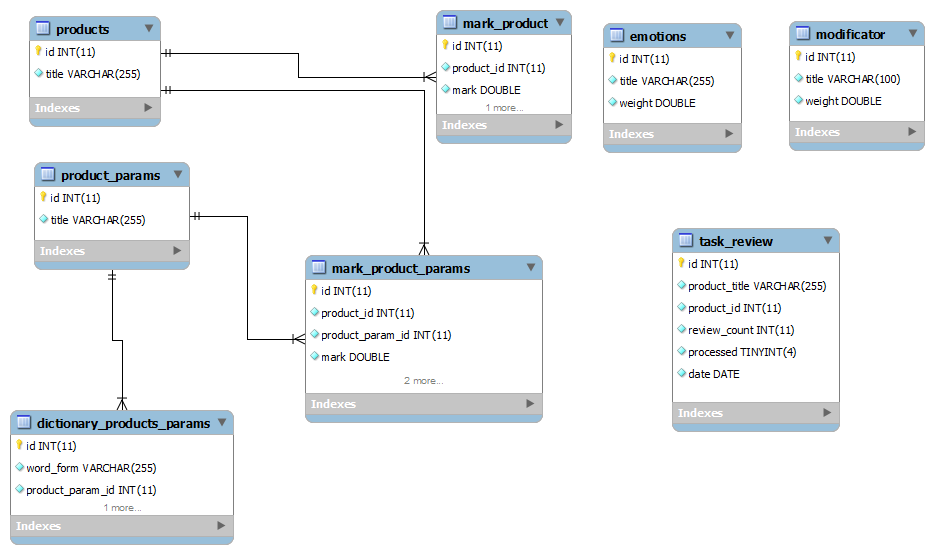
Как правило, современные средства создания Java-приложений разработаны для различных платформ: Linux, Solaris, Windows и MacOS. Важнейшее преимущество языка Java заключается в том, что приложение, написанное на основе данного языка, является независимым от платформы и архитектуры процессора, который будет выполнять алгоритм данного приложения. Главным звеном в данном процессе является виртуальная машина Java — это специальная программа, которая имеющийся откомпилированный независимый байт-код преобразует в набор инструкций для конкретного процессора. Программа должна быть предварительно установлена на компьютер, где планируется запуск приложения.

Язык Java является объектно-ориентированным и поставляется с достаточно объемной библиотекой классов. Библиотеки классов Java значительно упрощают разработку приложений, представляя в распоряжение программиста мощные средства решения стандартных задач. Подключение к базе данных осуществляется с помощью JDBC.

Выберем язык Java, технология JDBC позволяет легко взаимодействовать с СУБД MySQL, существует большой выбор фреймворков для разработки web — приложений.

### Описание структуры данных

Важнейшая цель разработки информационной модели заключается в выработке непротиворечивой интерпретации данных и взаимосвязей между ними. Это является необходимым для совместного использования и управления целостностью данных в рамках разрабатываемой системы. На рисунке 2.2. приведена информационная модель данных, в которой таблицы **emotions** и **modificator** содержат эмоционально окрашенные слова с оценками. Таблицы **mark\_product** и **mark\_product\_params** содержат оценки, каждая из которых вычислена по одному отзыву. Таблица **dictionary\_products\_params** содержит словоформы названия параметров продукта. Таблица **products** содержит список продуктов, для которых производились исследования. Таблица **product\_params** содержит список параметров продукта. Таблица **task\_review** сожержит задания на обработку отзывов из Яндекс маркета.

Рисунок 2.2. Информационная модель БД.

### Описание структуры программного продукта

Логически систему можно разделить на три части:

Первая часть — анализатор. Она предназначена для проведения анализа продуктов.

Вторая часть — редакторы. Она предназначена для редактирования словарей системы.

Третья часть — статистика. Она предназначена для отображения статистических данных по проведенным анализам.

Структура программы отображена на рисунке 2.3. в виде взаимодействия логических модулей программы.

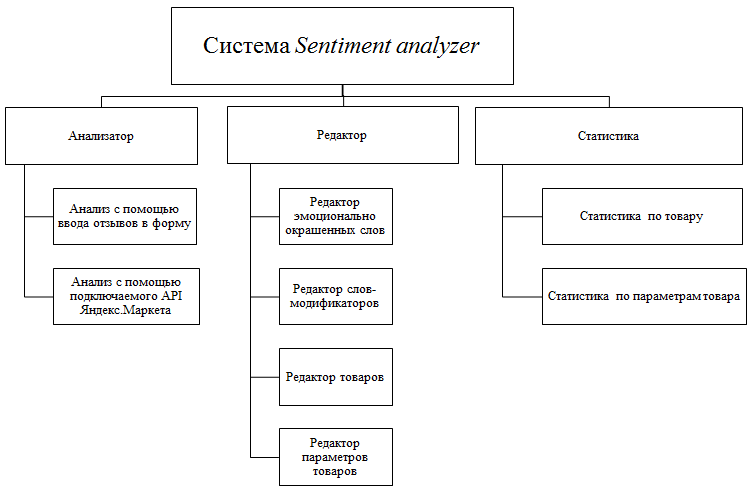


Рисунок 2.3. Структура программного продукта.

### Описание интерфейса пользователя

После загрузки программы открывается главная страница (рисунок 2.4.).

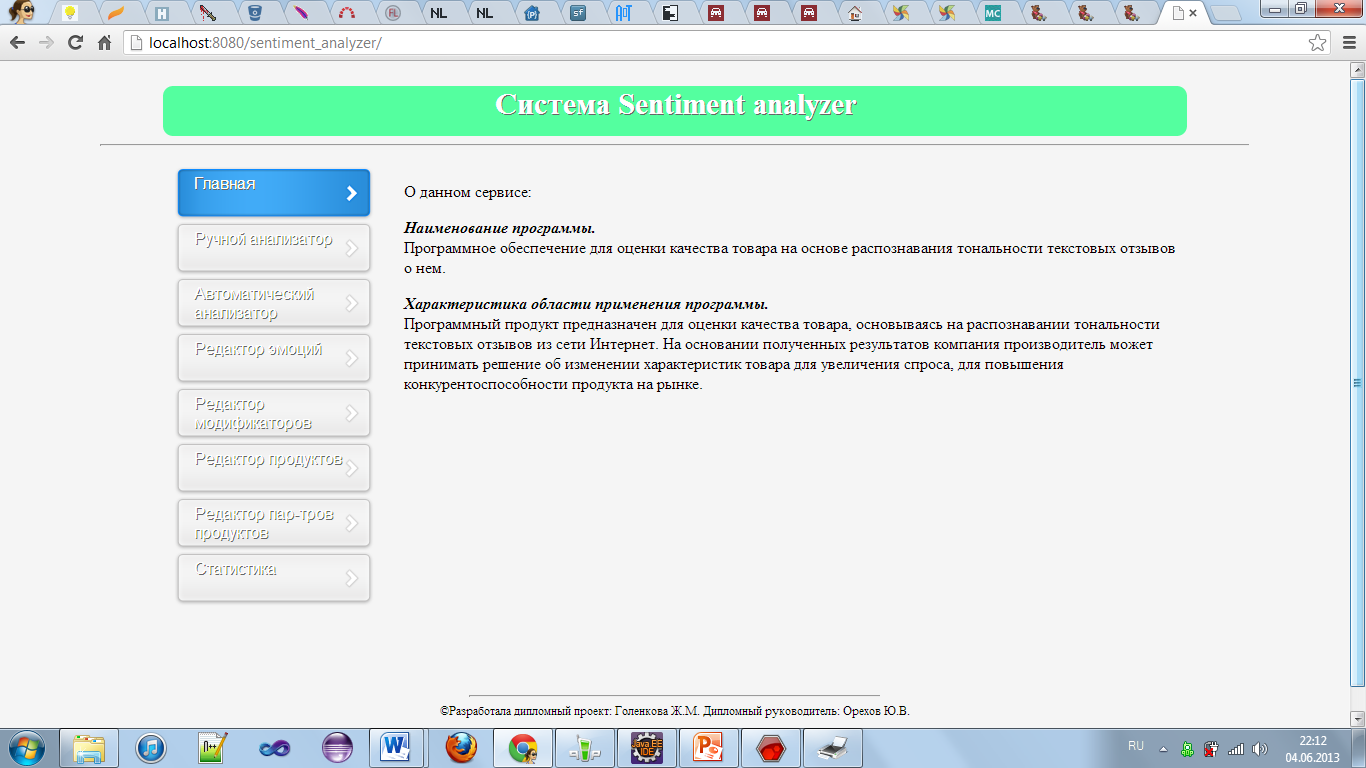
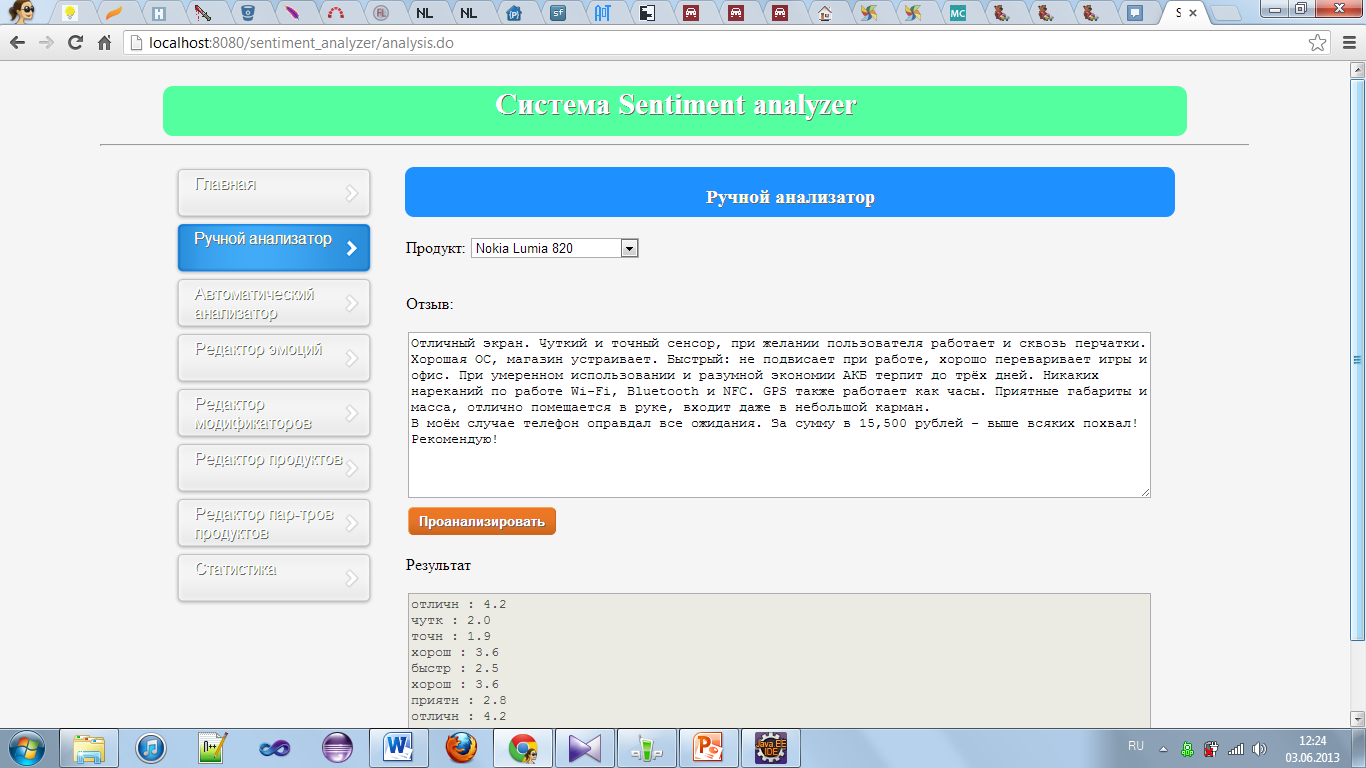
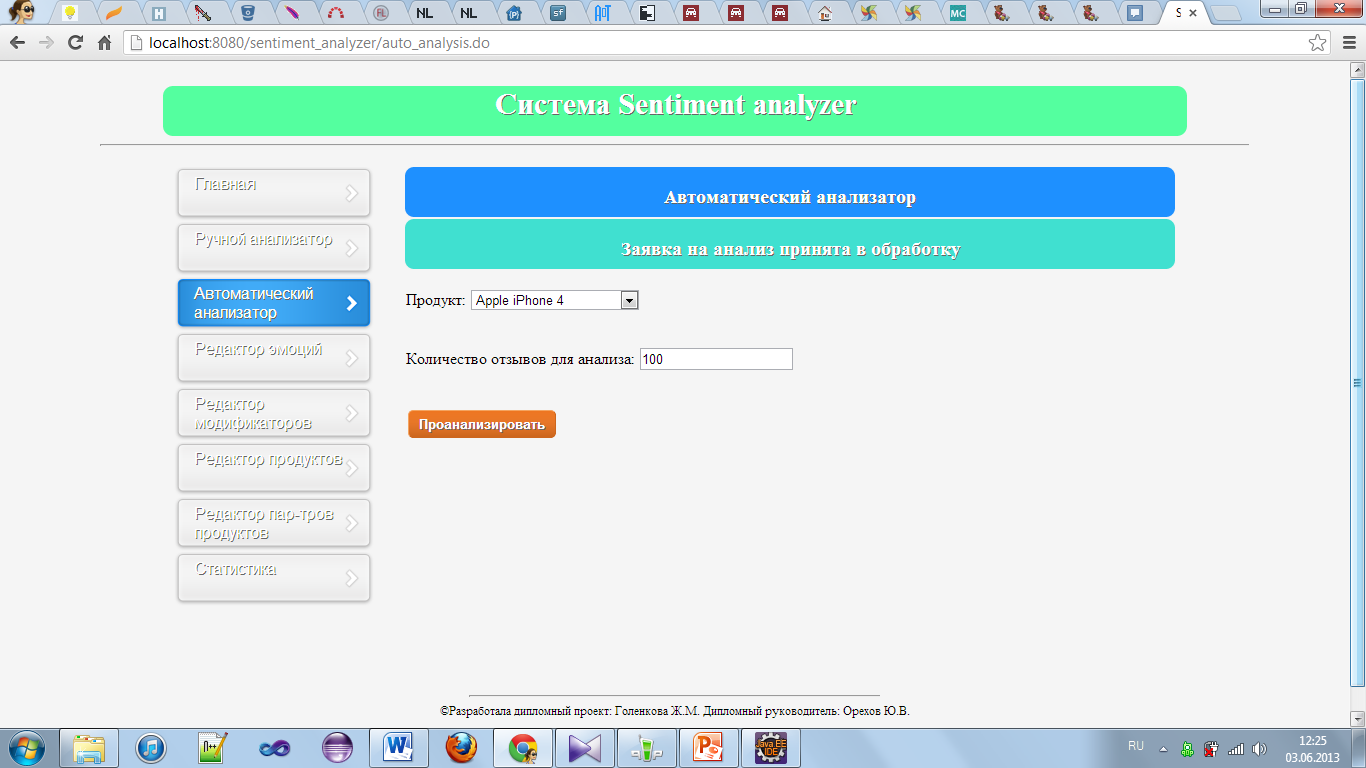


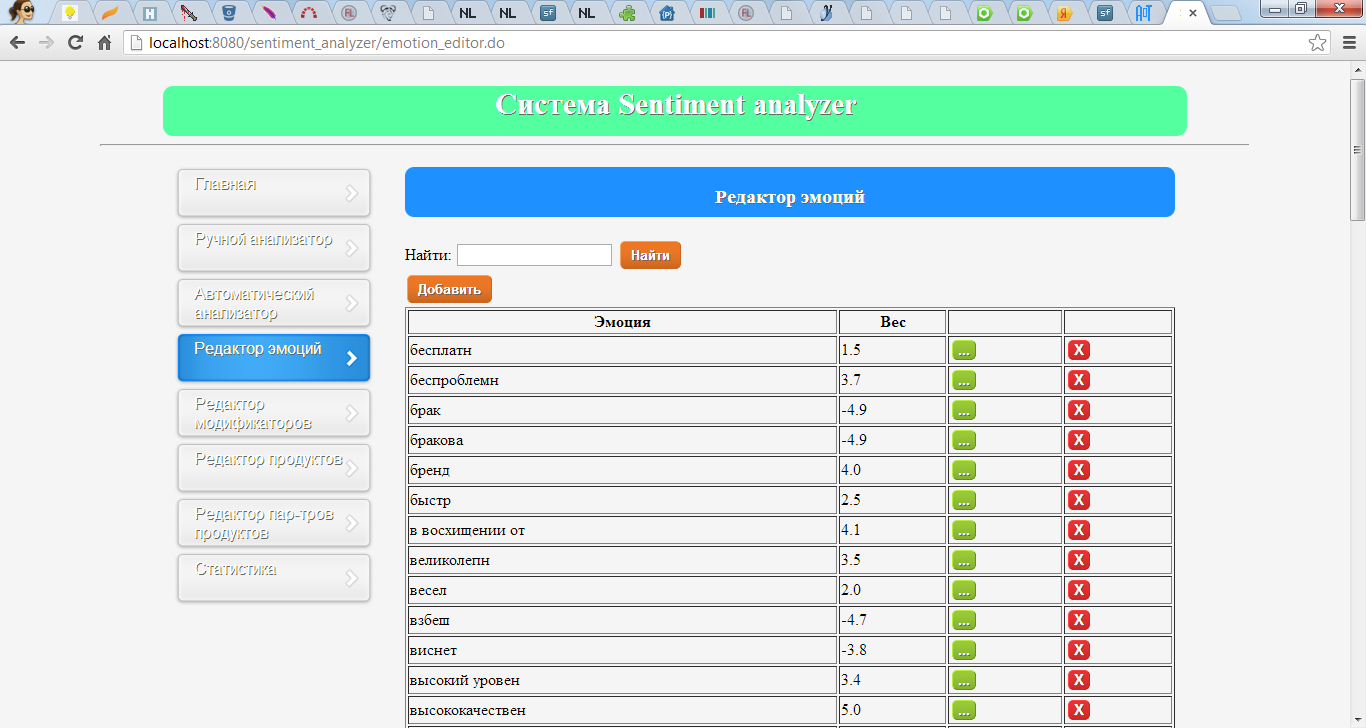
Рисунок 2.4. Главная страница программы.

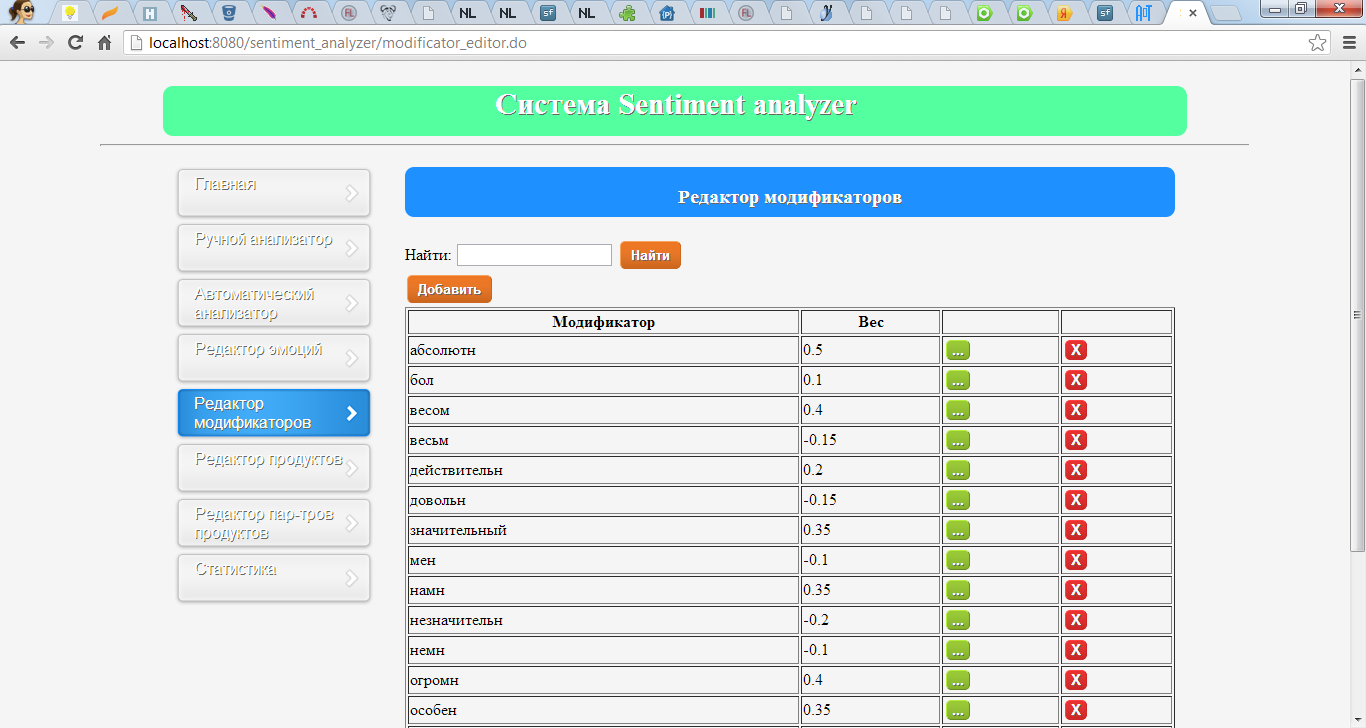
Слева расположено основное меню программы, при помощи которого, можно переходить в соответствующие разделы.

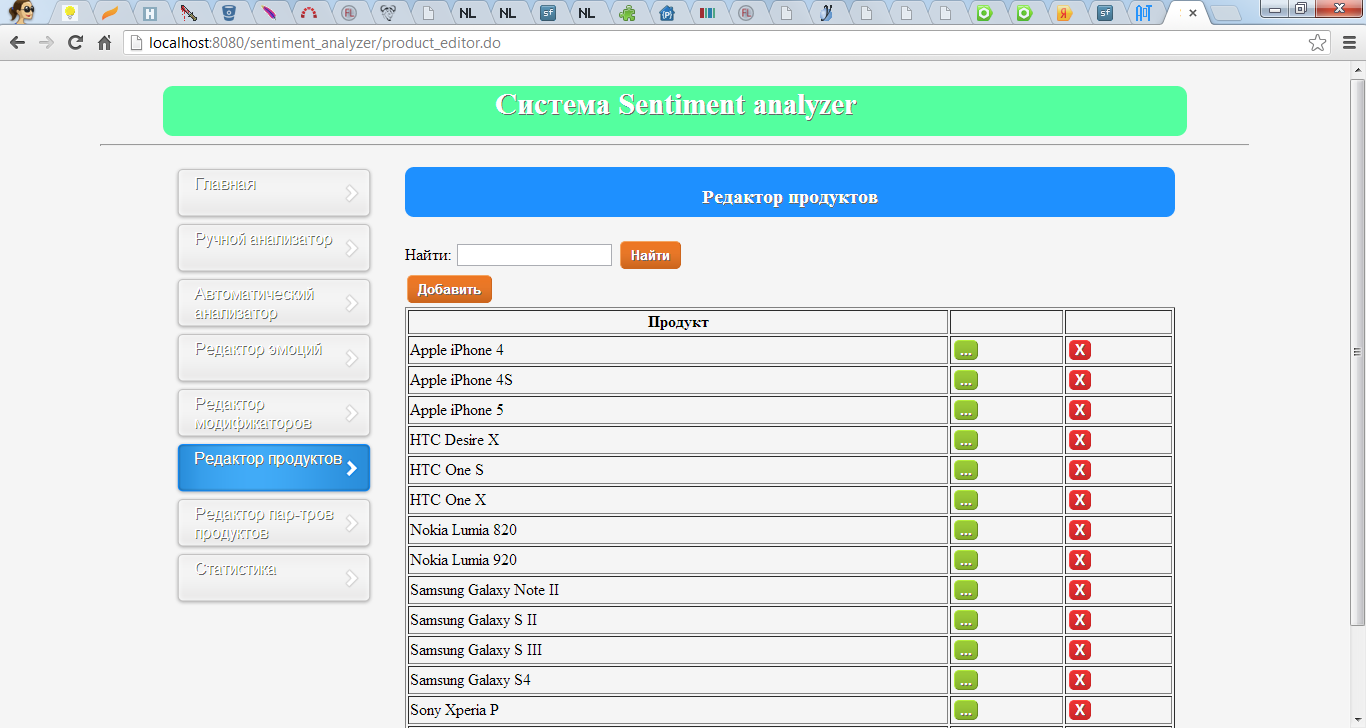
На следующих рисунках будут представлены соответствующие разделы программы.

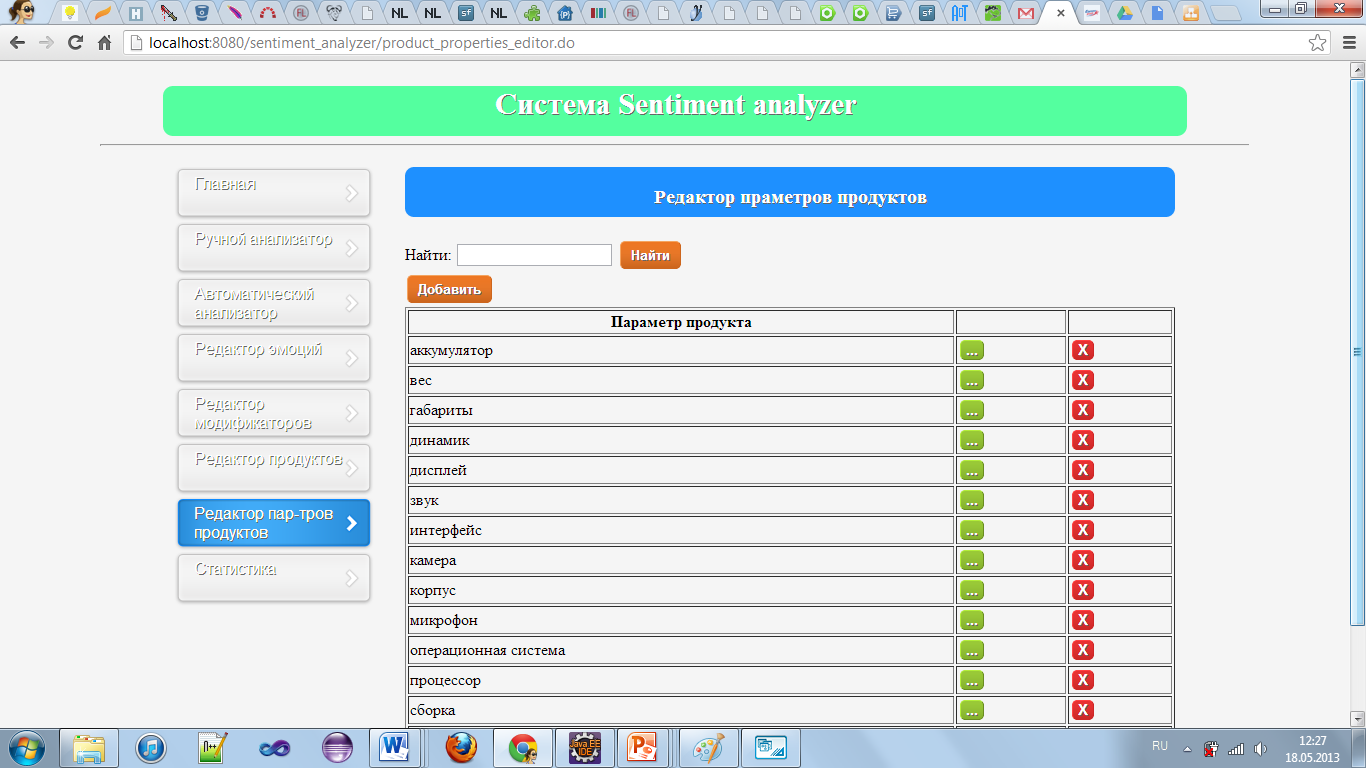
Рисунок 2.5. Ручной анализатор.

Рисунок 2.6. Автоматический анализатор.

Рисунок 2.7. Редактор эмоционально окрашенных слов.

Рисунок 2.8. Редактор слов-модификаторов.

Рисунок 2.9. Редактор продуктов.

 Рисунок 2.10. Редактор параметров продуктов.

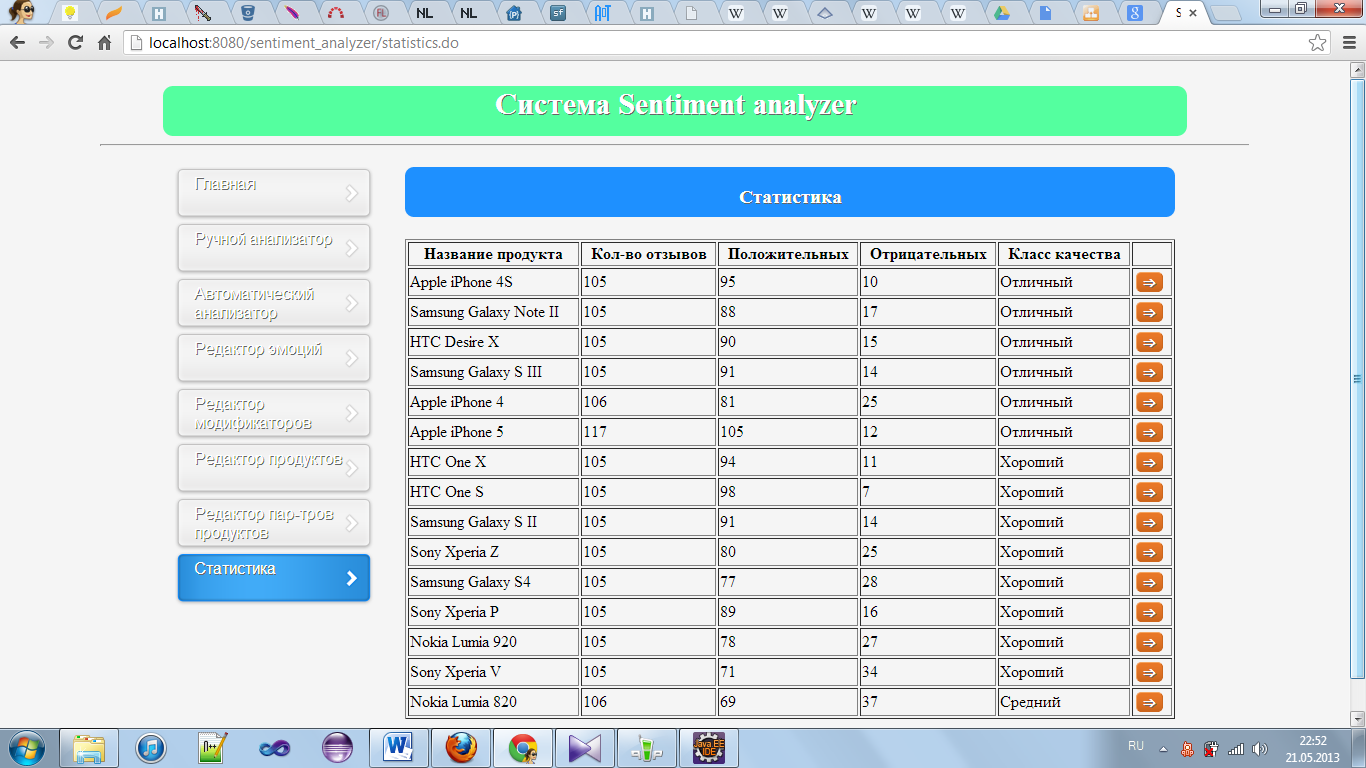


Рисунок 2.11. Статистика по товарам.

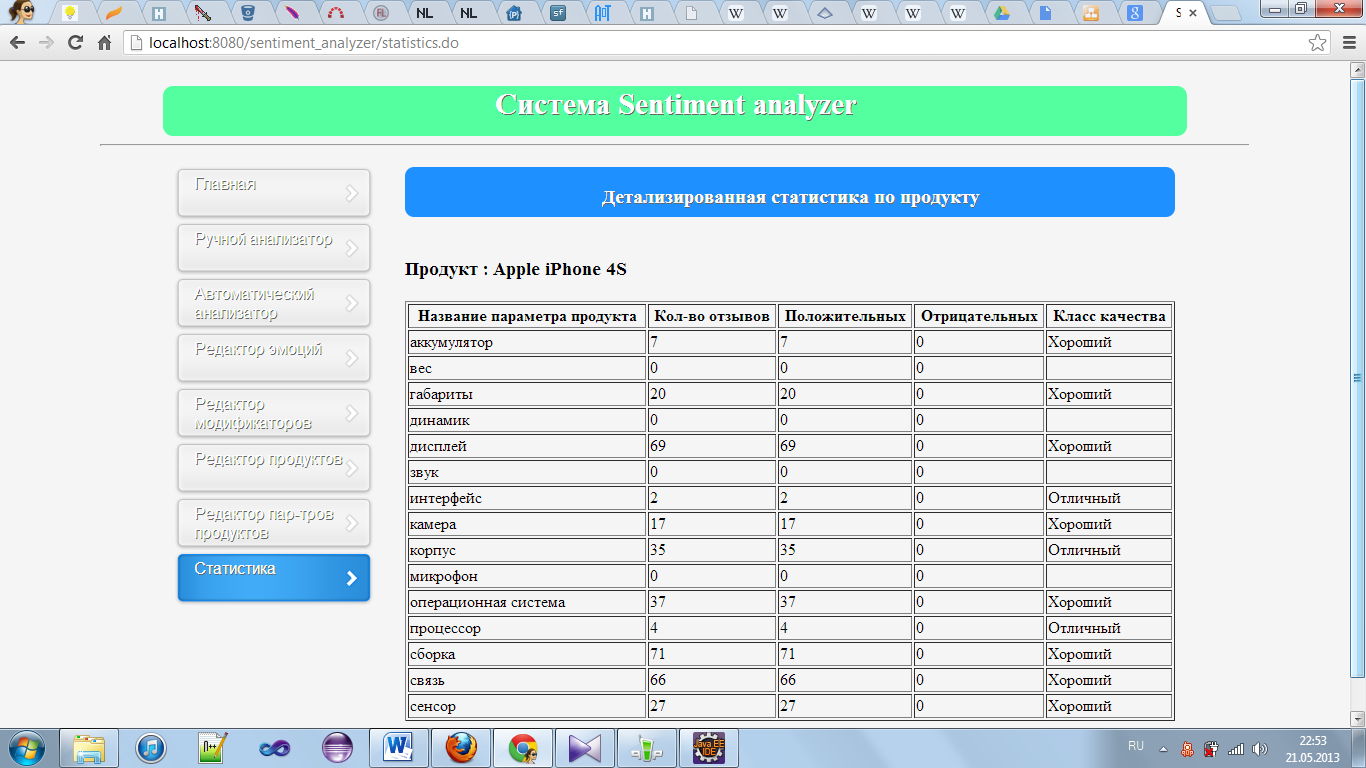


Рисунок 2.12. Статистика по параметрам товара.

## Тестирование ПО

Для проверки правильности работы программы рекомендуется, чтобы несколько компьютеров стояли в одном помещении и были подключены к локальной сети или к глобальной сети.

Процесс тестирования можно разделить на следующие этапы:

* Проверка в нормальных условиях (правильность работы программы для характерных данных);
* Проверка в экстремальных условиях:
  + работоспособность программы для граничных значений области изменения входных данных;
  + правильность работы программы при граничных объемах исходной информации (для большого числа данных или без данных вообще);
* Проверка в исключительных ситуациях.
* реакция программы на нулевые примеры (нулевое значение числовых переменных, пробелы для символьных и строковых величин).

**Тестирование в нормальных условиях**

Тестирование в нормальных условиях предполагает тестирование на основе данных и условий, которые характерны для реального функционирования программы. В данном случае нормальные условия это: наличие связи с базой данных, ввод корректных данных.

При проверке в нормальных условиях все функции программы работают правильно, и выдается достоверная информация.

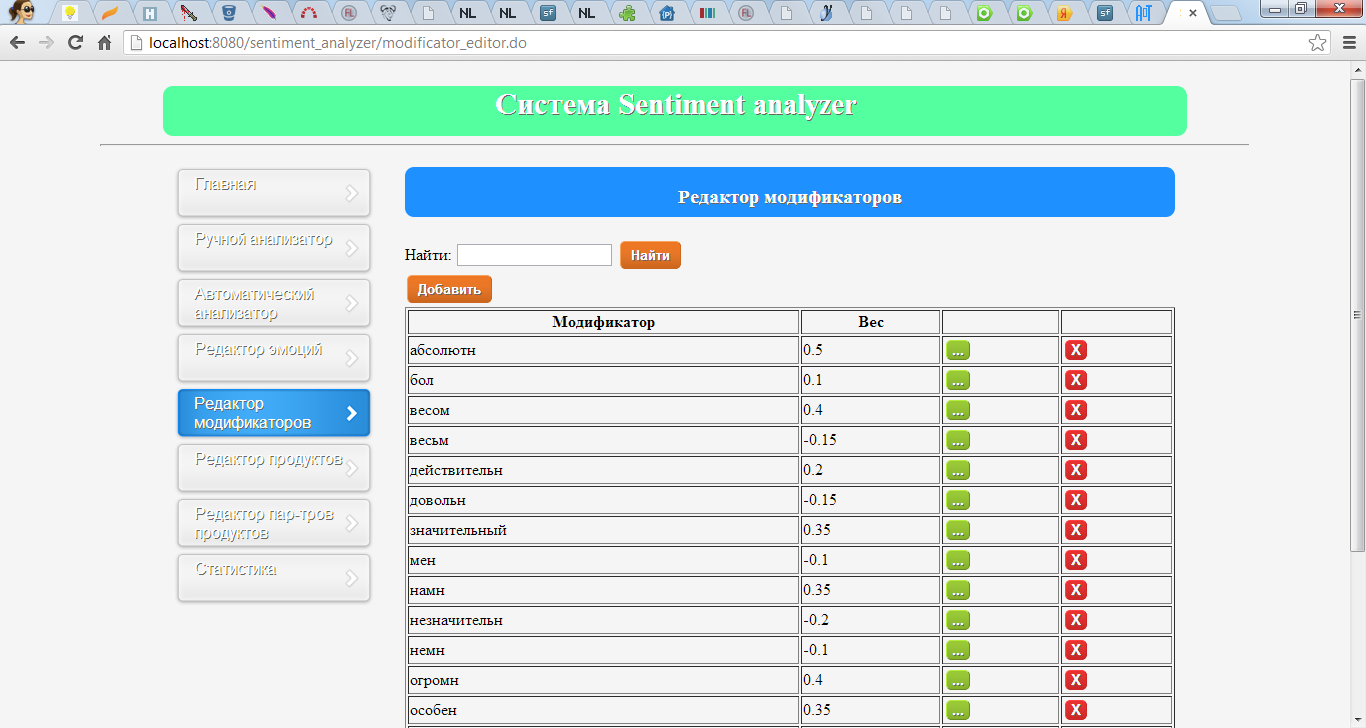


Рисунок 2.13. Правильная работа программы

**Тестирование в экстремальных условиях**

Правильность работы программы при граничных объемах исходных данных – слишком большое число записей или вообще нет ни одной. Трудно дать оценку на работоспособность программы при граничных объемах, т.к. база данных может содержать сотни тысяч записей и работать бесперебойно. Работа системы тестировалась, когда в БД имелось более 1500 записей – функционирование протекало удовлетворительно. При отсутствии связи с БД программа реагировала адекватно, выводя об этом сообщение пользователю.

В программе производится контроль всех вводимых параметров. При некорректном вводе выдается соответствующее предупреждение и операция, приведшая к этому, отменяется.

На этом этапе тестирования проводилась проверка соответствия типов данных и вводимой информации. Было верно определено, соответствует ли тип вводимой информации необходимому типу. В случаях, когда это условие не выполнялось, выдавалось сообщение об ошибке. Таким образом, пресекался неверный ввод данных.

Например, в поле имеющее тип число нет возможности внести данные типа строка. При вводе строковой информации в числовые поля, программа сопоставляет строку с числовым форматом, если сопоставление не удалось, то выводится сообщение о некорректном вводе данных.

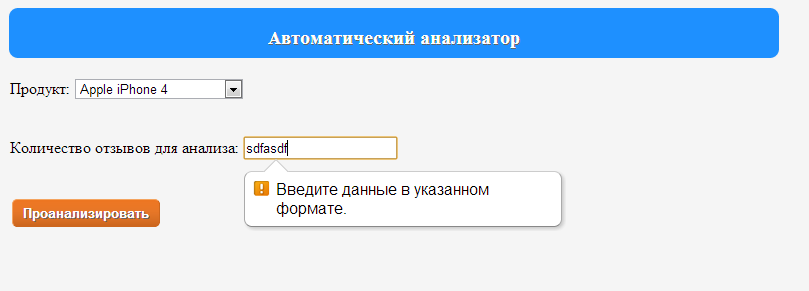


Рисунок 2.14. Сообщение о неверном формате данных

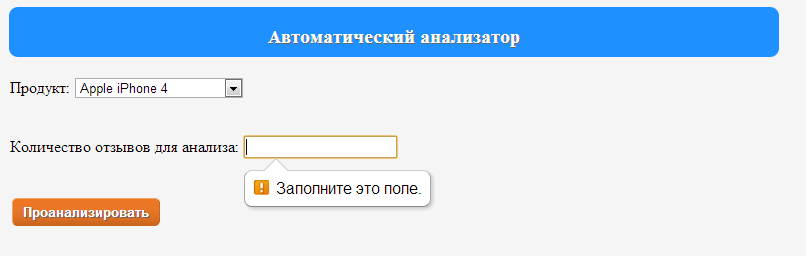


Рисунок 2.15. Сообщение о незаполненном поле

**Тестирование в исключительных ситуациях**

Исключительная ситуация возникает при попытке ввода дублирующей информации по полям указанным в базе данных как уникальные. Программа определяет такие ситуации, выводит сообщение, что такая информация уже существует, и отменяет ее запись.

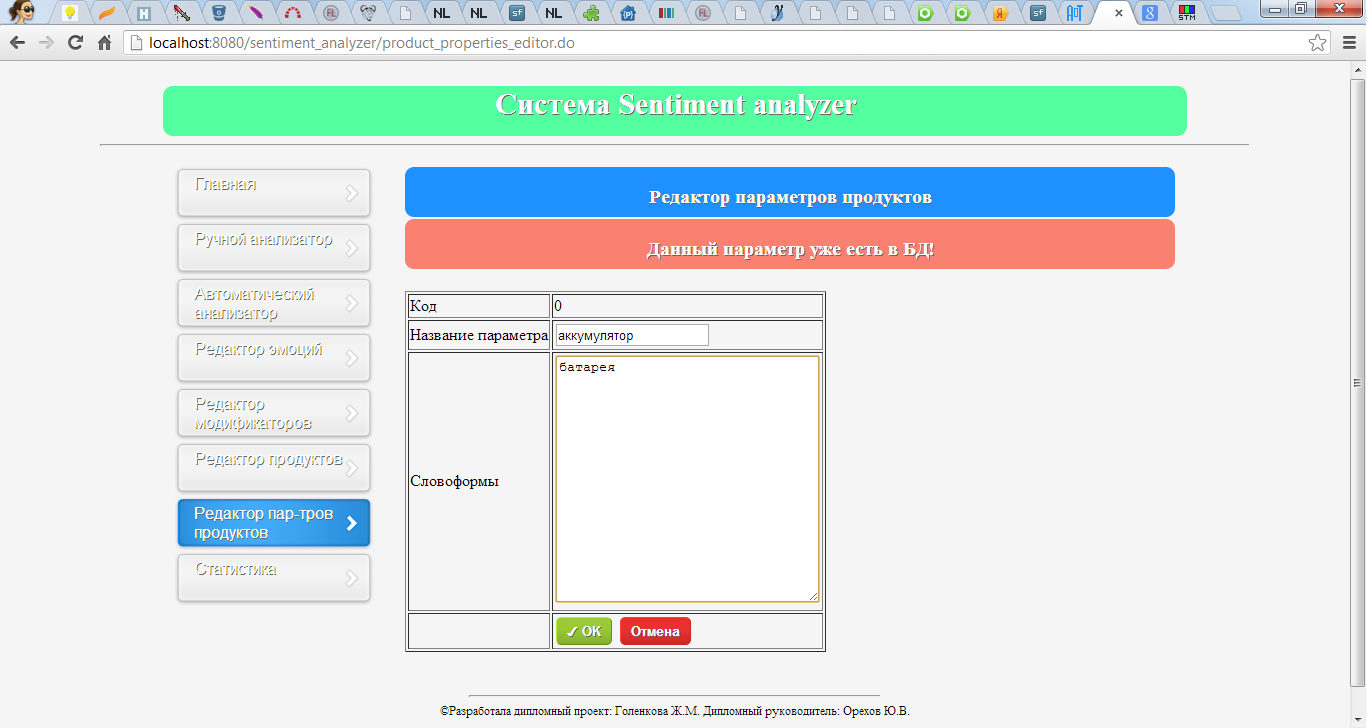


Рисунок 2.16. Сообщение дублирование информации

**Оценка полноты проверки программы**

Программный продукт был проверен на выполнение всех заложенных в него функций.

Система не позволяет пользователю вводить большинство недопустимых значений, что существенно упрощает процесс тестирования, минимизирует риск возникновения исключительных ситуаций и позволяет достаточно полно проверить работоспособность программного продукта в нормальных и экстремальных условиях.

Полное тестирование не проводится, ввиду его большого объема. Некоторые недоработки может определить только эксплуатация.

## Результаты экспериментальных исследований и их анализ

Для исследований результатов вводятся данные контрольного примера и проверяется работоспособность на данных контрольного примера и сравниваются с результатами ручной обработки данных.

Анализ результатов экспериментальных исследований, проведенных в двух точках, дает положительный результат работоспособности программы. Однако некоторые недоработки может определить только эксплуатация, но в рамках этих условий, руководствуясь результатами тестирования можно сказать об удовлетворительной работе программного продукта.

Результат экспериментальных исследований по части автоматического анализа тональности текстовых отзывов проводились на достаточно большом объеме данных (порядка 1500 записей) и показали приемлемые результаты работы.

Для оценки точности классификации будем использовать показатель «точности классификации», который вычисляется по формуле:

где – числа верно классифицированных отзывов для каждого из пяти классов, *NT* – число верно классифицированных отзывов, *NF* – число неверно классифицированных отзывов.

## Программная документация

### Техническое задание

**1. Введение.**

***1.1. Наименование программы.***

Программное обеспечение для оценки качества товара на основе распознавания тональности текстовых отзывов о нем.

***1.2. Характеристика области применения программы.***

Программный продукт предназначен для оценки качества товара, основываясь на распознавании тональности текстовых отзывов из сети Интернет. На основании полученных результатов компания производитель может принимать решение об изменении характеристик товара для увеличения спроса, для повышения конкурентоспособности продукта на рынке.

**2. Основания для разработки.**

Основанием для разработки данного программного обеспечения служит задание научного руководителя выпускной квалификационной работы.

**3. Назначение разработки.**

***3.1. Функциональное назначение программы.***

Система должна оценивать качество товара, основываясь на распознавании тональности текстовых отзывов из сети Интернет. Система должна автоматически обрабатывать текстовый отзыв, выделяя в нем заданные параметры и эмоционально окрашенные слова к ним относящиеся. Должен проводиться расчет оценки отзыва. На основе полученных данных система должна определять класс качества товара и параметров товара. Система должна вести учет проанализированных товаров и полученных оценок качества.

Полезность программы будет заключаться в том, что минимизируются затраты времени на ручной анализ отзывов, на составление анкет и их анализ, а также на опрос пользователей.

***3.2. Эксплуатационное назначение программы.***

Данная система предназначается для использования в организациях занимающихся продажей и разработкой мобильных телефонов.

**4. Требования к программе.**

***4.1. Требования к функциональным характеристикам программы.***

Данная система должна обеспечивать:

* автоматизация обработки текстового отзыва;
* выделение эмоционально окрашенных слов в текстовом отзыве;
* выделение заданных параметров, относящихся к товару;
* расчет оценки отзыва;
* определение класса качества товара и параметров товара в соответствии с полученными оценками отзывов;
* учет проанализированных товаров и полученных оценок качества;
* создание, редактирование, удаление тональных словарей.
  1. ***Требования к надежности программы.***

Система должна обеспечивать полностью устойчивое функционирование (за исключением сбоев из-за неисправности системного ПО или отсутствия подключения к сети Internet) и не нарушать целостность вычислительной системы и/или системного ПО. Входные данные должны проверяться на корректность. Сбои в работе системы и длительное время выполнения являются недопустимыми.

* 1. ***Требования к входным и выходным данным***

В качестве входных данных системы будут использоваться:

* + текстовый отзыв;
  + заданные параметры товара;
  + словарь эмоционально окрашенных слов;
  + словарь слов-модификаторов.

В качестве выходных данных система должна выдавать:

* + класс качества товаров;
  + класс качества параметров товара.

***4.4. Требования к составу и параметрам технических и программных средств.***

Рекомендуемые аппаратные требования к компьютеру, где будет располагаться серверная часть программного продукта:

* архитектура x86/x64;
* процессор - Pentium IV 2,5 ГHz;
* объем оперативной памяти – не ниже 512 Mb;
* объем жесткого диска не менее 100Gb;
* сетевая карта.

Необходимое программное обеспечение для работы серверной части программного продукта:

* MySQL Server 5.5;
* JDK 7;
* рекомендованная операционная система MS Windows 7;
* web — сервер apache tomcat 7.

Аппаратные и программные требования к компьютеру, где будет использоваться клиентская часть — любые, где сможет запуститься один из современных интернет браузеров с поддержкой java script и HTML5. А также для использования клиентской части программного продукта необходимы монитор, клавиатура и мышь.

***4.5. Требования к информационной и программной совместимости.***

Клиентская часть программного продукта должна корректно функционировать под управлением ОС MS Windows XP/7;

**5. Состав и требования к программной документации.**

***5.1. Предварительный состав программной документации.***

Программная документация, поставляемая с программой, должна включать:

* «Техническое задание».
* «Руководство программиста».
* «Руководство пользователя».

***5.2. Специальные требования к программной документации.***

Полнота освещение всех аспектов, затронутых при проектировании и реализации используемых в программном продукте методов и подходов.

### Руководство программиста

*Назначение и условия применения программы*

Программный продукт предназначен для оценки качества товара, основываясь на распознавании тональности текстовых отзывов из сети Интернет. На основании полученных результатов компания производитель может принимать решение об изменении характеристик товара для увеличения спроса, для повышения конкурентоспособности продукта на рынке.

В программе реализованы функции:

* автоматизация обработки текстового отзыва;
* выделение эмоционально окрашенных слов в текстовом отзыве;
* выделение заданных параметров, относящихся к товару;
* расчет оценки отзыва;
* определение класса качества товара и параметров товара в соответствии с полученными оценками отзывов;
* учет проанализированных товаров и полученных оценок качества;
* создание, редактирование, удаление тональных словарей.

Для работы серверной части программы необходима следующая аппаратная конфигурация:

* архитектура x86/x64;
* процессор - Pentium IV 2,5 ГHz;
* объем оперативной памяти – не ниже 512 Mb;
* объем жесткого диска не менее 100Gb;
* сетевая карта.

Необходимое программное обеспечение для работы серверной части программного продукта:

* MySQL Server 5.5;
* JDK 7;
* рекомендованная операционная система MS Windows 7;

*Характеристики программы*

Логически систему можно разделить на три части:

Первая часть — анализатор. Она предназначена для проведения анализа продуктов.

Вторая часть — редакторы. Она предназначена для редактирования словарей системы.

Третья часть — статистика. Она предназначена для отображения статистических данных по проведенным анализам.

*Входные и выходные данные*

В качестве входных данных системы используются:

* + текстовый отзыв;
  + заданные параметры товара;
  + словарь эмоционально окрашенных слов;
  + словарь слов-модификаторов.

В качестве выходных данных система должна выдавать:

* + класс качества товаров;
  + класс качества параметров товара.

*Структура программы*

Структура программы отображена на рисунке 2.17. в виде взаимодействия логических модулей программы.

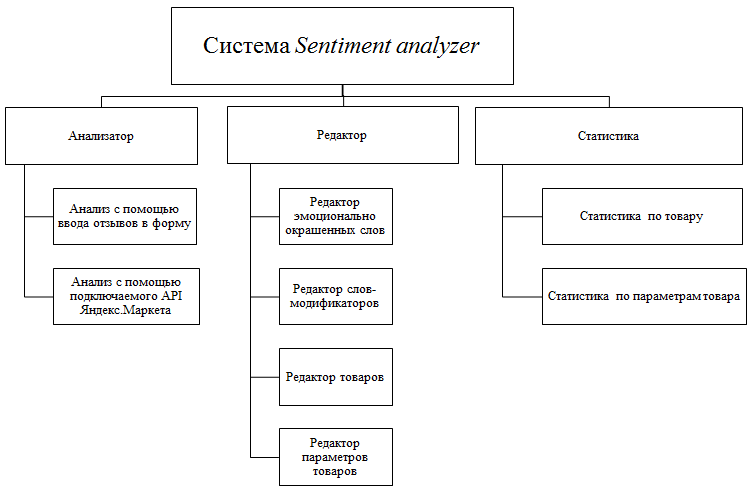


Рисунок 2.17. Структура программного продукта

**Алгоритмическая часть**

Приведем краткое описание всех разделов:

* *Ручной анализатор* - данный раздел позволяет обрабатывать по одному отзыву;
* *Автоматический анализатор* - данный раздел предназначен для добавления задач на обработку отзывов по заданному продукту в Яндекс маркете;
* *Редакторы* – данный раздел предназначен для добавления, редактирования и удаления сущностей. В данный раздел входят:
  1. редактор слов с эмоциональной окраской;
  2. редактор слов-модификаторов;
  3. редактор продуктов;
  4. редактор параметров продуктов.
* *Статистика* - данный раздел предназначен для просмотра статистических данных о проведенных анализах продуктов.

Рассмотрим более подробно структуру взаимодействия классов внутри модулей. Взаимодействие между серверной и клиентской частями приведено на схеме работы фреймворка struts1 (рисунке 2.18.).

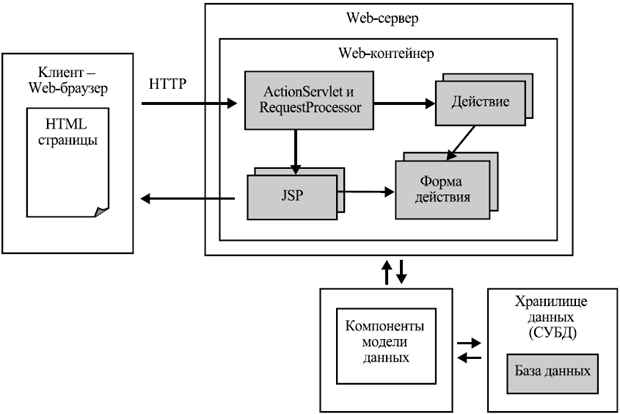


Рисунок 2.18. Схема работы фреймворка struts1.

Связка между классами Действие, классами Форма действия и jsp файлами производится в конфигурационном файле фреймворка – struts-config.xml

***JSP файлы.***

*index* – главная страница;

*head* – общая часть страниц, подключается во все остальные jsp файлы.

*header* – общая часть страниц, подключается во все остальные jsp файлы.

*footer* – общая часть страниц, подключается во все остальные jsp файлы.

*analysis* – раздел ручного анализатора

*auto\_analysis* – раздел автоматического анализатора

*emotion* – раздел редактора слов с эмоциональной окраской

*modificator* – раздел редактора слов-модификаторов

*product* – раздел редактора продуктов

*product\_properties* – раздел редактора продуктов

*statistics* – раздел статистики

***Классы Действия***

* + - 1. *AnalysisAction –*  обрабатывает действия раздела ручного анализатора. Наследуется от *ActionBase.*

**Методы класса:**

*public ActionForward execute()*– переопределенный метод базового класса Action фреймворка struts1.

* + - 1. *AutoAnalysis – обрабатывает действия раздела автоматического анализатора.* Наследуется от *ActionBase.*

**Методы класса:**

*public ActionForward execute()*– переопределенный метод базового класса Action фреймворка struts1.

* + - 1. *StatisticsAction –* обрабатывает действия раздела статистики. Наследуется от *ActionBase.*

**Методы класса:**

*public ActionForward execute()*– переопределенный метод базового класса Action фреймворка struts1.

*private void statistics()* – подготовка и передача в jsp информации для общей статистики.

*private void detailStatistics()* – подготовка и передача в jsp информации для статистики по параметрам выбранного продукта.

Каждый класс действия из раздела редакторов наследуется и реализует абстрактные методы, определенные в базовом классе *EditorsActionBase*.

*EmotionEditorAction* – обрабатывает действия раздела редактирования слов с эмоциональной окраской.

*ModificatorEditorAction* – обрабатывает действия раздела редактирования слов-модификаторов.

*ProductsEditorAction* – обрабатывает действия раздела редактирования продуктов.

*ProductsPropertiesEditorAction* – обрабатывает действия раздела редактирования параметров продуктов.

***Базовые классы действия системы.***

1. *EditorsActionBase* – базовый абстрактный класс для обработки действий создать/отредактировать/удалить в разделах личного кабинета, содержит описание абстрактных методов, которые реализуются в конечных классах действия. Наследуется от базового класса *ActionBase*.

**Методы класса:**

*public ActionForward execute()*– переопределенный метод базового класса Action фреймворка struts1.

*protected abstract doFilter –* поиск по имеющимся записям

*protected abstract getForward*– возвращает перенаправление внутри раздела после действия

*protected abstract void update()*– сохраняет введенную информацию в БД

*protected abstract void cancel()*– сброс редактора и возврат в список сущностей

*protected abstract void edit()*– заполняет редактор параметрами выбранной сущности

*protected abstract void newItem()*– заполняет редакторы необходимыми общими данными для заведения новой сущности.

*protected abstract void delete()*– удаляет сущность из БД.

1. *ActionBase* – базовый класс, содержащий статичные структуры данных, которые используются в конечных классах действия.

**Поля класса:**

*protected Setup setup* – настройки всего приложения

***Классы Формы действий***

Класс формы действия отвечает за передачу параметров между jsp файлами и классами действия. Классы имеют вид структуры данных содержащих приватные поля и публичные методы для получения и установки значений.

1. *AnalysisActionForm* – описывает параметры в разделе ручного анализатора.

**Поля класса:**

*private String review* – отзыв;

*private int productId* – код продукта;

1. *AutoAnalysisActionForm* – описывает параметры в разделе автоматического анализатора.

**Поля класса:**

*private int productId* – код продукта ;

*private int reviewCount* – количество отзывов, которое необходимо проанализировать;

1. *EmotionActionForm* – описывает параметры в разделе редактирования слов с эмоциональной окраской.

**Поля класса:**

*private double weight* – вес эмоционального слова;

1. *EmotionActionForm* – описывает параметры в разделе редактирования слов с эмоциональной окраской.

**Поля класса:**

*private double weight* – вес эмоционального слова;

1. *ModificatorActionForm* – описывает параметры в разделе редактирования слов-модификаторов.

**Поля класса:**

*private double weight* – вес модификатора;

1. *ProductsActionForm* – описывает параметры в разделе редактирования продуктов.
2. *ProductsPropertiesActionForm* – описывает параметры в разделе редактирования параметров продукта.

**Поля класса:**

*private String wordForm* – список словоформ для параметра;

1. *StatisticsActionForm* – описывает параметры в разделе статистики.

**Поля класса:**

*private String productTitle* – название продукта;

*private int productTitle* – код продукта

1. *BaseActionForm* – базовый класс, который расширяют все классы формы действия, и содержит в себе общие поля.

**Поля класса:**

*private int id* – код сущности;

*private String title* – название сущности;

*private String command* – команда( удалить/сохранить/выбран элемент).

***Классы Модели данных***

Класс модели данных имеет вид структуры данных, которая используется в серверной части программы и описывает поля конкретной таблицы. Содержит в себе поля (столбцы таблицы) и методы для установки и получения значений.

1. *Emotion* – описывает структуру таблицы *emotions*.
2. *Modificator* – описывает структуру таблицы *modificator*
3. *Product* – описывает структуру таблицы *products*.
4. *ProductProperties – описывает структуру таблицы product\_params*
5. *ProductPropertiesWordForm – описывает структуру таблицы dictionary\_products\_params*
6. *ProductPropertyStatistic – описывает структуру таблицы mark\_product\_params*
7. *ProductStatistic – описывает структуру таблицы mark\_product*
8. *TaskReview – описывает структуру таблицы task\_review*
9. *IdTitle* – базовая структура, которую расширяют остальные модели.
10. *Id* – базовая структура, которую расширяют остальные модели и базовый класс IdTitle.

***Классы взаимодействия с БД***

Класс взаимодействия с БД содержит методы для работы с конкретной таблицей в БД.

1. *EmotionDAO* – взаимодействие с таблицей *emotions*.
2. *ModificatorDAO* – взаимодействие с таблицей *modificator*.
3. *ProductDAO* – взаимодействие с таблицей *products*.
4. *ProductPropertiesDAO – взаимодействие с таблицей product\_params*
5. *ProductPropertiesWordFormDAO - взаимодействие с таблицей dictionary\_products\_params*
6. *ProducPropertyStatisticDAO - взаимодействие с таблицей mark\_product\_params*
7. *ProductStatisticDAO - взаимодействие с таблицей mark\_product*
8. *TaskReviewDAO - взаимодействие с таблицей task\_review*
9. *AbstractDAO* – содержит базовые универсальные методы для работы с таблицами, а также описание абстрактных методов, которые реализуются в каждом конкретном классе взаимодействия с таблицей.

**Методы класса:**

*public void update()* – универсальный метод обновления сущности в таблице вызывает абстрактный метод *updateImpl*;

*protected abstract void updateImpl* – каждый класс-наследник должен реализовать метод для своей таблицы;

*public B getById()*– универсальный метод, который возвращает сущность из таблицы по ее коду;

*protected abstract B getFromRS* – абстрактный метод, реализуемый в классе-наследнике для своей таблицы, возвращает модель данных, формируемую из результата запроса к БД;

*public List<B> getList* – возвращает список сущностей в таблице;

*public void delete()* – универсальный метод удаления сущности из таблицы по ее коду.

**Кофигурационные файлы серверной части программы**

1. *sentiment\_analyzer.properties* – глобальные настройки сервеной части. Содержит в себе настройки подключения к БД.
2. *jaas.config* – настройки авторизации, указывается какой класс отвечает за авторизацию пользователей.
3. *log4j.properties* – настройки системы логирования.
4. *web.xml* – настройки встроенного сервера tomcat. Также в нем прописываются роли пользователей и ресурсы, которые может посещать пользователь системы в зависимости от своей роли.

Подсистема обработки задач анализа отзывов в Яндекс маркете состоит из обработчика задач, анализатора отзывов, реализации API для работы с Яндекс маркетом.

*TaskReviewExecuter* – обработчик задач анализа отзывов

*YandexMarketAnalyzer* – анализатор отзывов из яндексмаркета

*YandexMarketAPI* – реализация API для работы с яндекс маркетом.

Диаграмма классов представлена на рисунке 2.19.

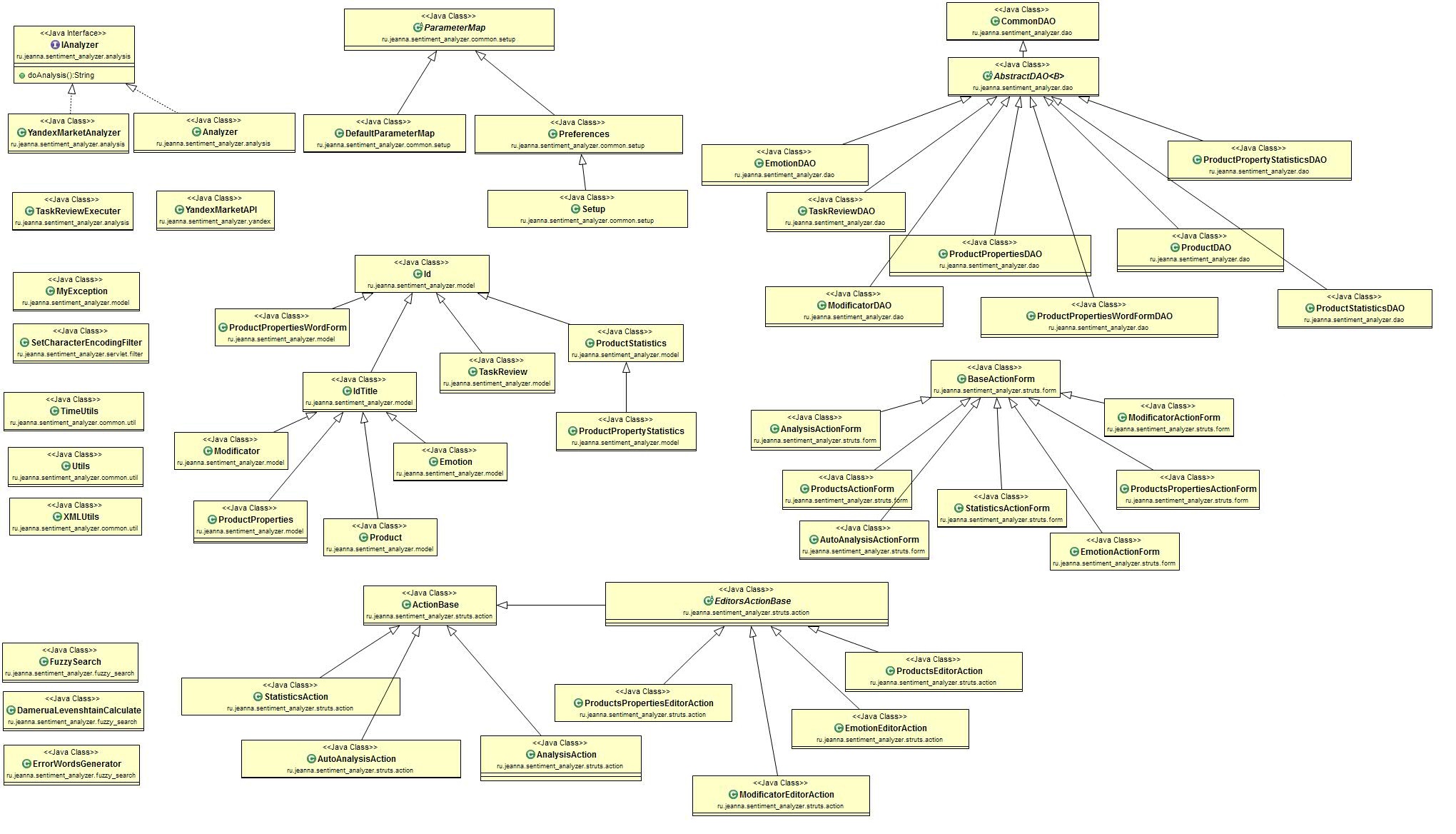
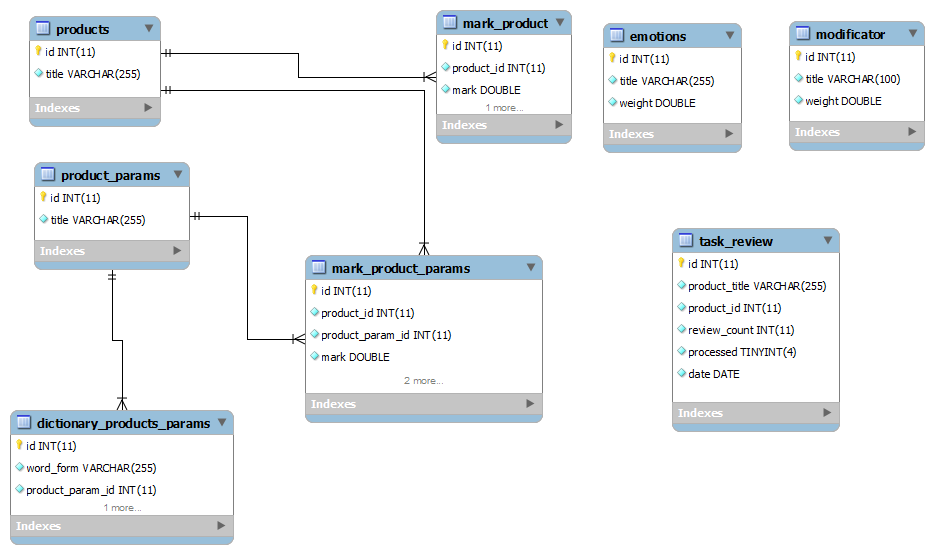


Рисунок 2.19. Диаграмма классов

**Структура базы данных**

Для удобства реализации и наглядного представления физического хранения информации была разработана модель предметной области(рисунок 2.20.).

Рисунок 2.20. Информационная модель БД

**emotions** и **modificator** содержат эмоционально окрашенные слова с оценками

**mark\_product** и **mark\_product\_params** содержат оценки, каждая из которых вычислена по одному отзыву

**dictionary\_products\_params** содержит словоформы названия параметров продукта

**products** содержит список продуктов, для которых производились исследования

**product\_params** содержит список параметров продукта

**task\_review** сожержит задания на обработку отзывов из Яндекс маркета

*Сообщения программисту*

В программе сообщения программисту выводятся при помощи системы логирования log4 в лог файл /log/sentiment\_analyzer.log. Типы сообщений и их формат настраиваются в конфигурационном файле log4j.properties.

### Руководство пользователя

*Назначение и условия применения программы*

Программный продукт предназначен для оценки качества товара, основываясь на распознавании тональности текстовых отзывов из сети Интернет. На основании полученных результатов компания производитель может принимать решение об изменении характеристик товара для увеличения спроса, для повышения конкурентоспособности продукта на рынке.

В программе реализованы функции:

* автоматизация обработки текстового отзыва;
* выделение эмоционально окрашенных слов в текстовом отзыве;
* выделение заданных параметров, относящихся к товару;
* расчет оценки отзыва;
* определение класса качества товара и параметров товара в соответствии с полученными оценками отзывов;
* учет проанализированных товаров и полученных оценок качества;
* создание, редактирование, удаление тональных словарей.

Аппаратные и программные требования к компьютеру, где будет использоваться клиентская часть — любые, где сможет запуститься один из современных интернет браузеров с поддержкой java script и HTML5. А также для использования клиентской части программного продукта необходимы монитор, клавиатура, мышь и подключение к сети Internet.

*Выполнение программы*

Для запуска серверной части программы необходимо прописать настройки в файле sentiment\_analyzer.properties, либо оставить настройки по умолчанию. Далее необходимо развернуть базу данных из файла sentiment\_analyzer.sql на сервере MySQL. После успешного запуска серверной части в браузере обратиться на указанный в конфигурации URL сервера по умолчанию localhost:8080/sentiment\_analyzer/index.jsp.

На рисунке 2.21. показана главная страница программы.

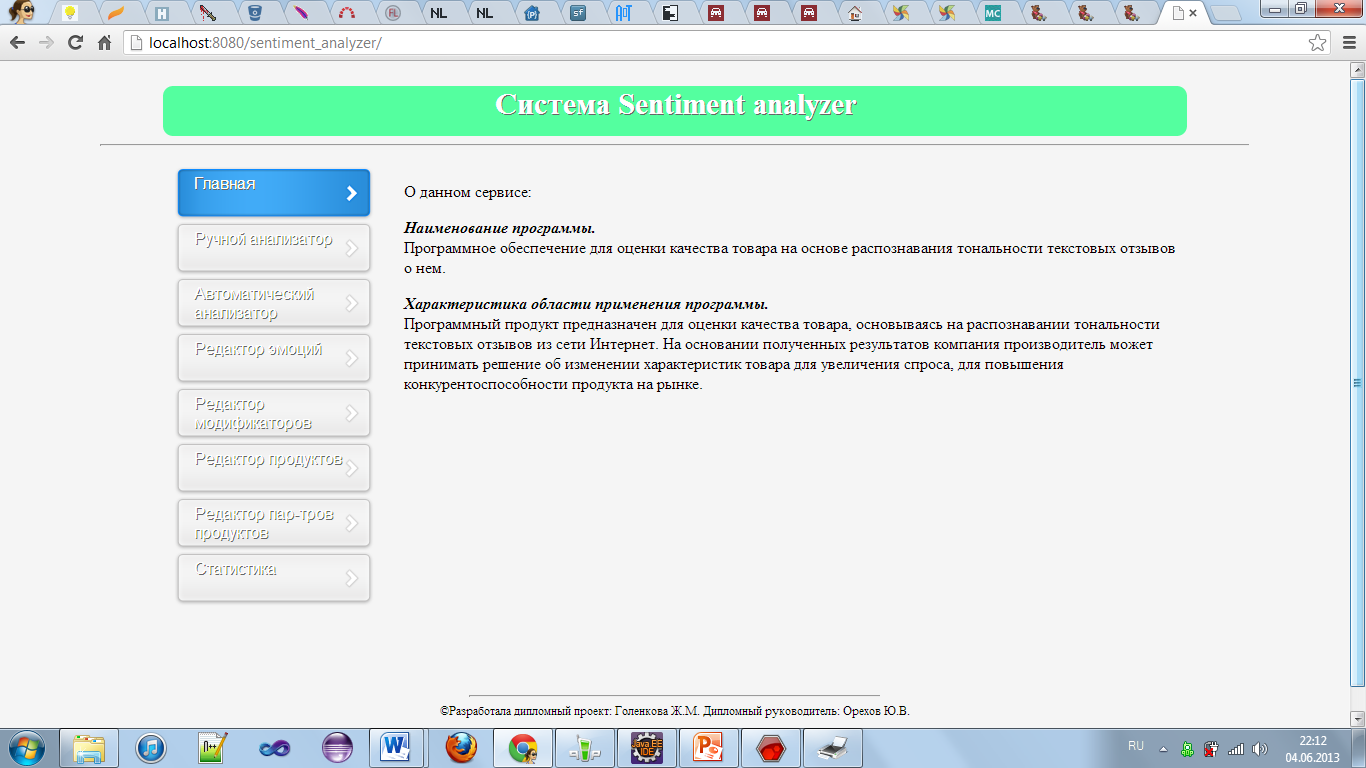


Рисунок 2.21. Главная страница программы

Слева расположено основное меню программы, при помощи которого, можно переходить в соответствующие разделы. Далее опишем интерфейс каждого раздела.

**Ручной анализатор**

Данный раздел позволяет обрабатывать по одному отзыву. Для этого необходимо выбрать продукт, отзыв о котором будет анализироваться, и заполнить поле отзыв. После чего нажать на кнопку «Проанализировать». В результате анализа отзыва в поле «Результаты» будут выведены результаты анализа.

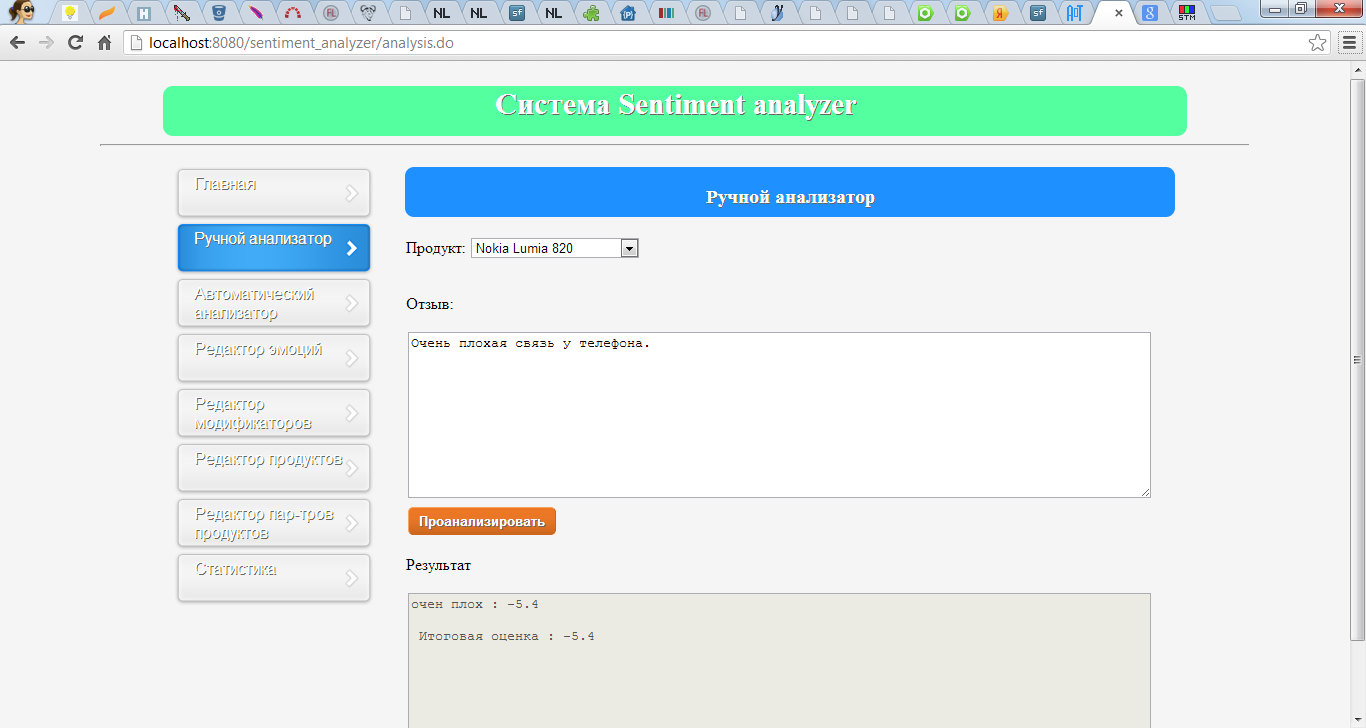


Рисунок 2.22. Ручной анализатор

**Автоматический анализатор**

Данный раздел предназначен для добавления задач на обработку отзывов по заданному продукту в Яндекс маркете. Для того чтобы добавить задачу на обработку отзывов необходимо выбрать продукт и указать количество отзывов, которое следует обработать.

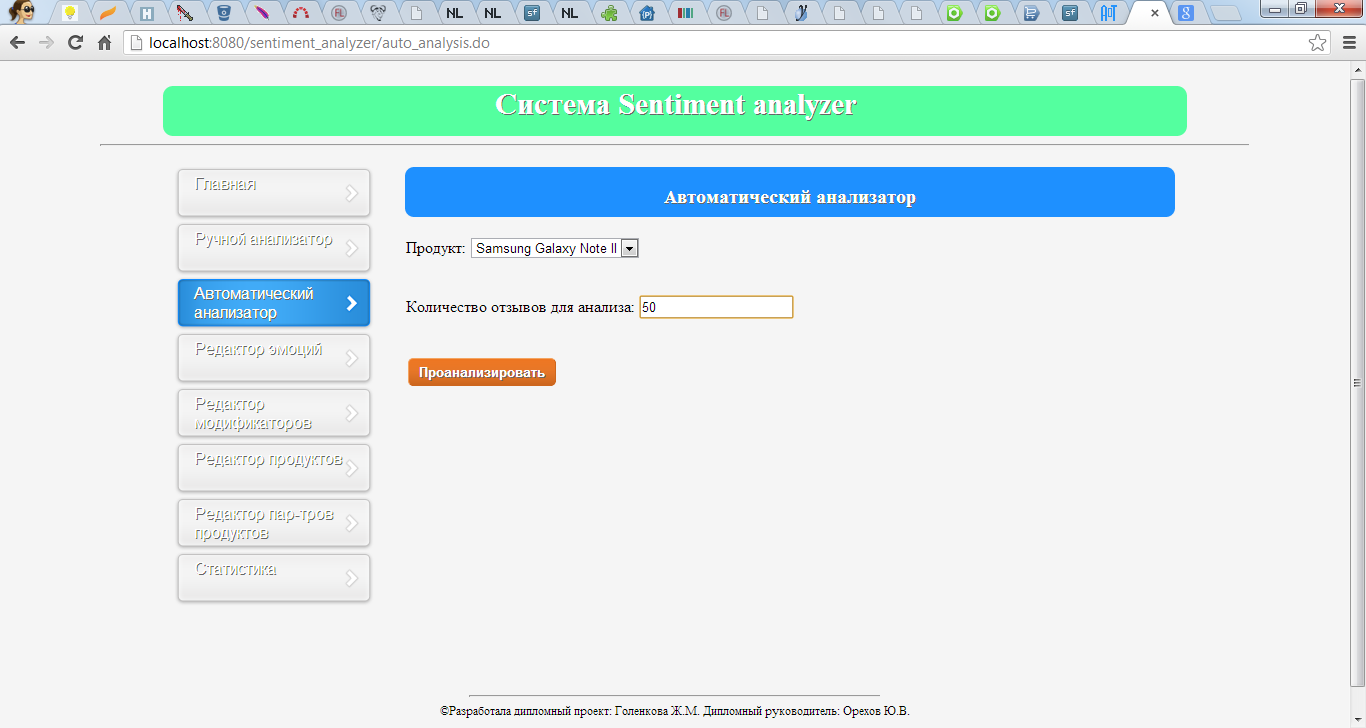


Рисунок 2.23. Автоматический анализатор

**Редакторы**

Интерфейсы всех редакторов очень схожи, поэтому рассмотрим интерфейс редактора на примере редактора слов с эмоциональной окраской.

В верхней части раздела располагается фильтр, предназначенный для поиска сущностей. Для того чтобы добавить новый отзыв необходимо нажать на кнопку «Добавить», после чего нужно заполнить соответствующие поля и нажать на кнопку «ОК» чтобы сохранить изменения. Для редактирования записи необходимо нажать на кнопку «…», а для удаления записи – «Х».

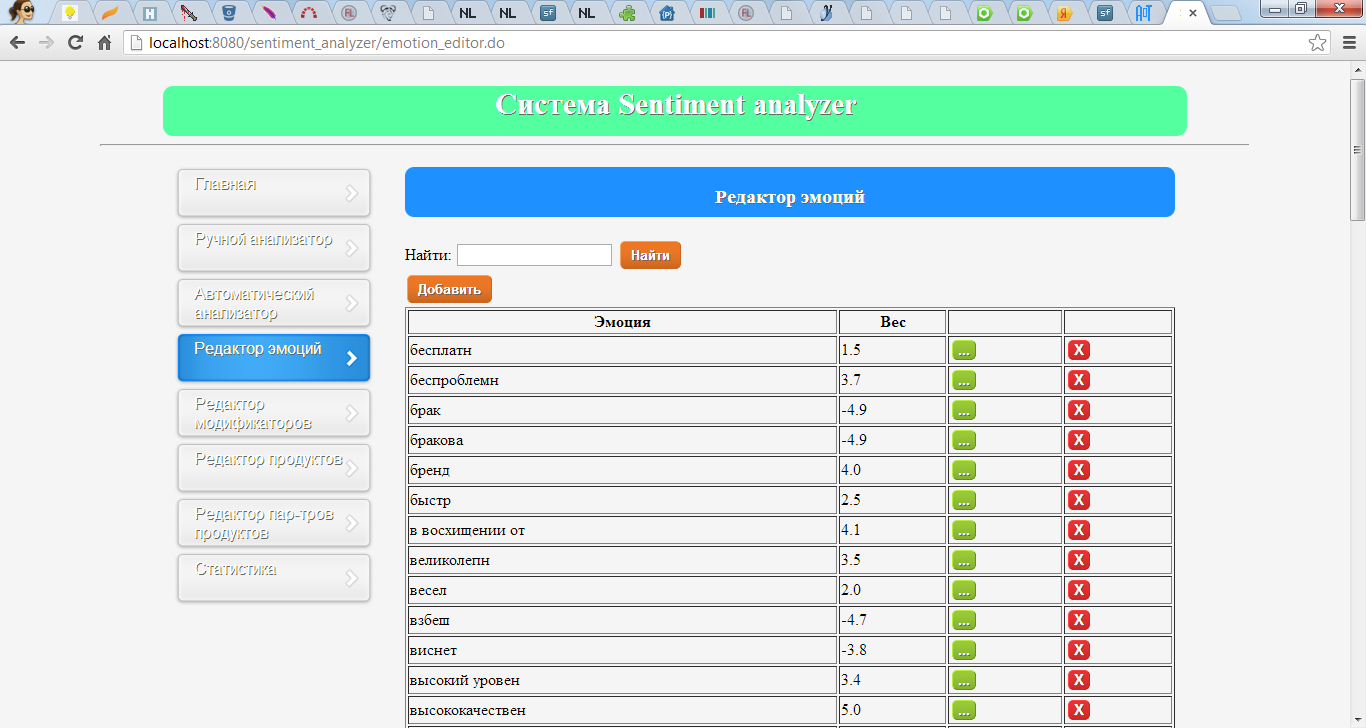


Рисунок 2.24. Список сущностей

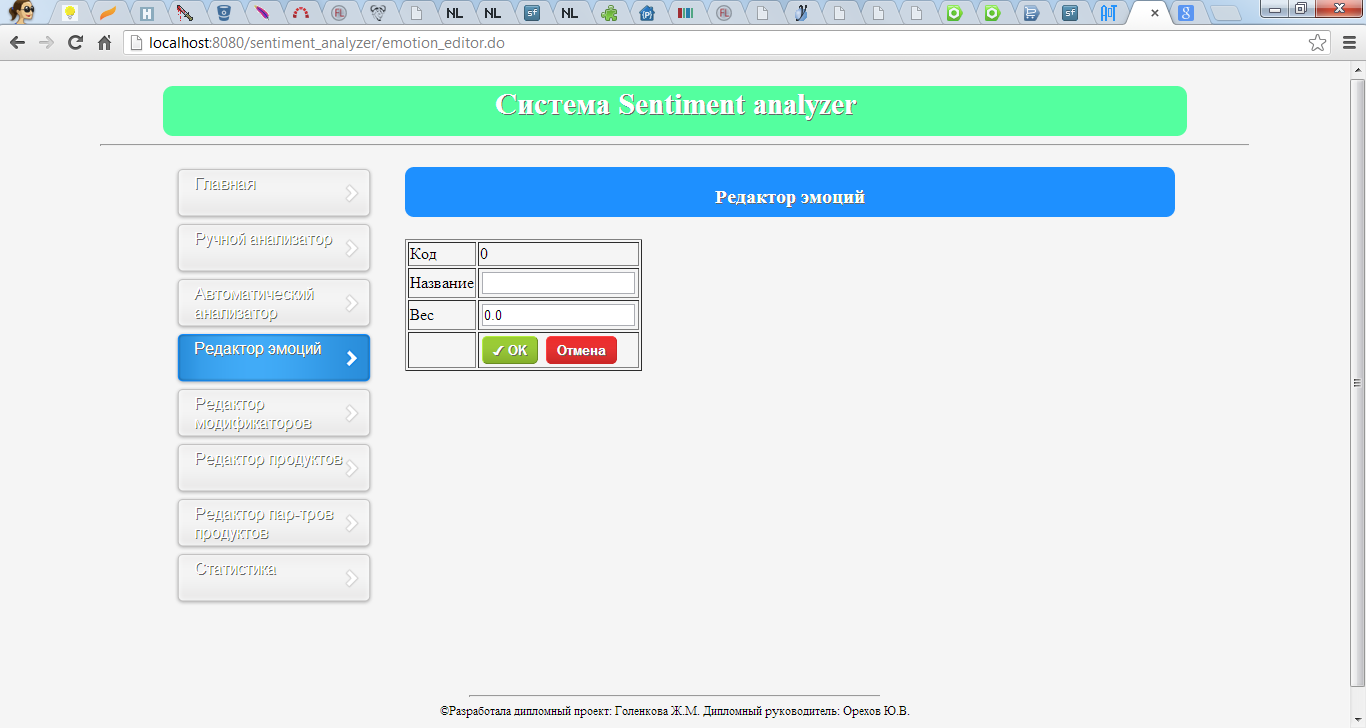


Рисунок 2.25. Редактор

**Статистика**

Данный раздел предназначен для просмотра статистических данных о проведенных анализах продуктов. Есть 2 вида статистики. В первом виде можно просмотреть информацию по продуктам. Во втором – информацию по параметрам выбранного продукта. Для того чтобы из первого вида переключиться во второй, необходимо нажать на кнопку «=>» около продукта.

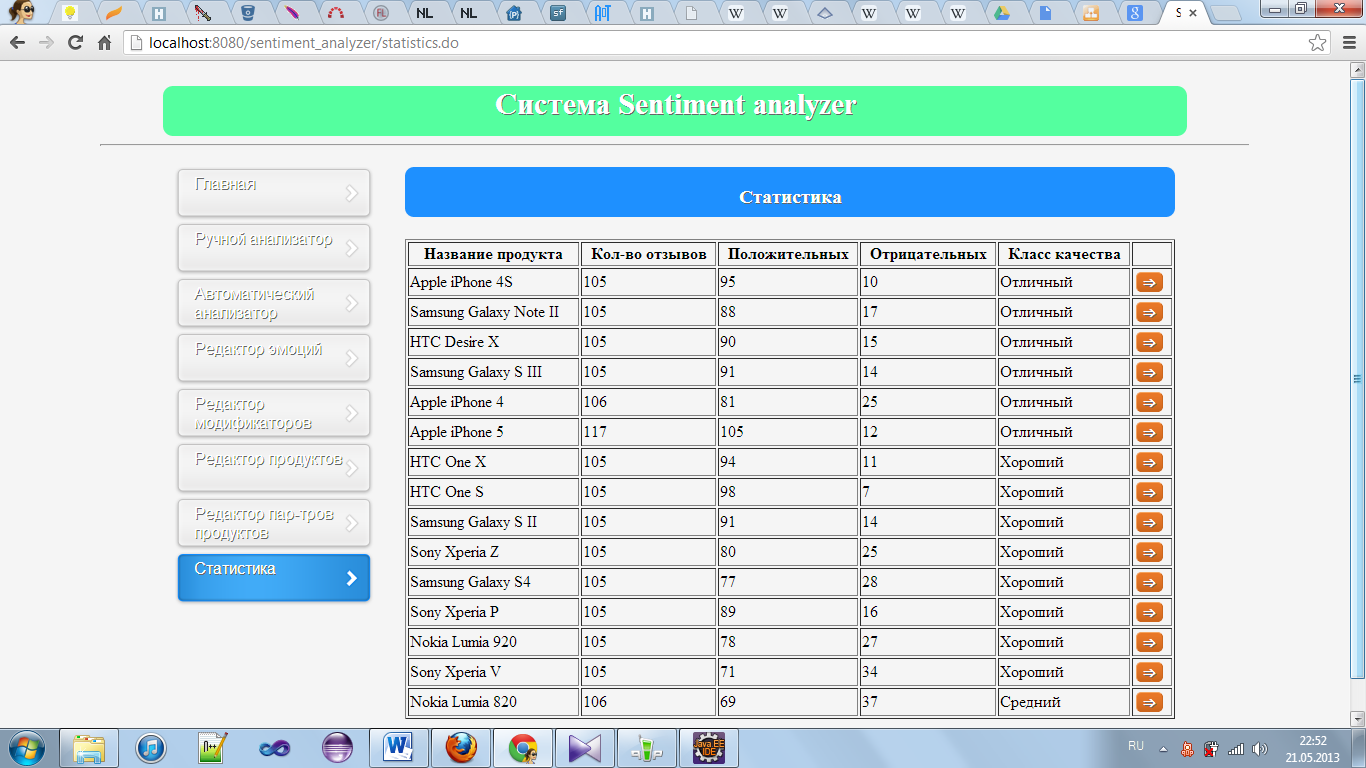


Рисунок 2.26. Статистика по продуктам.

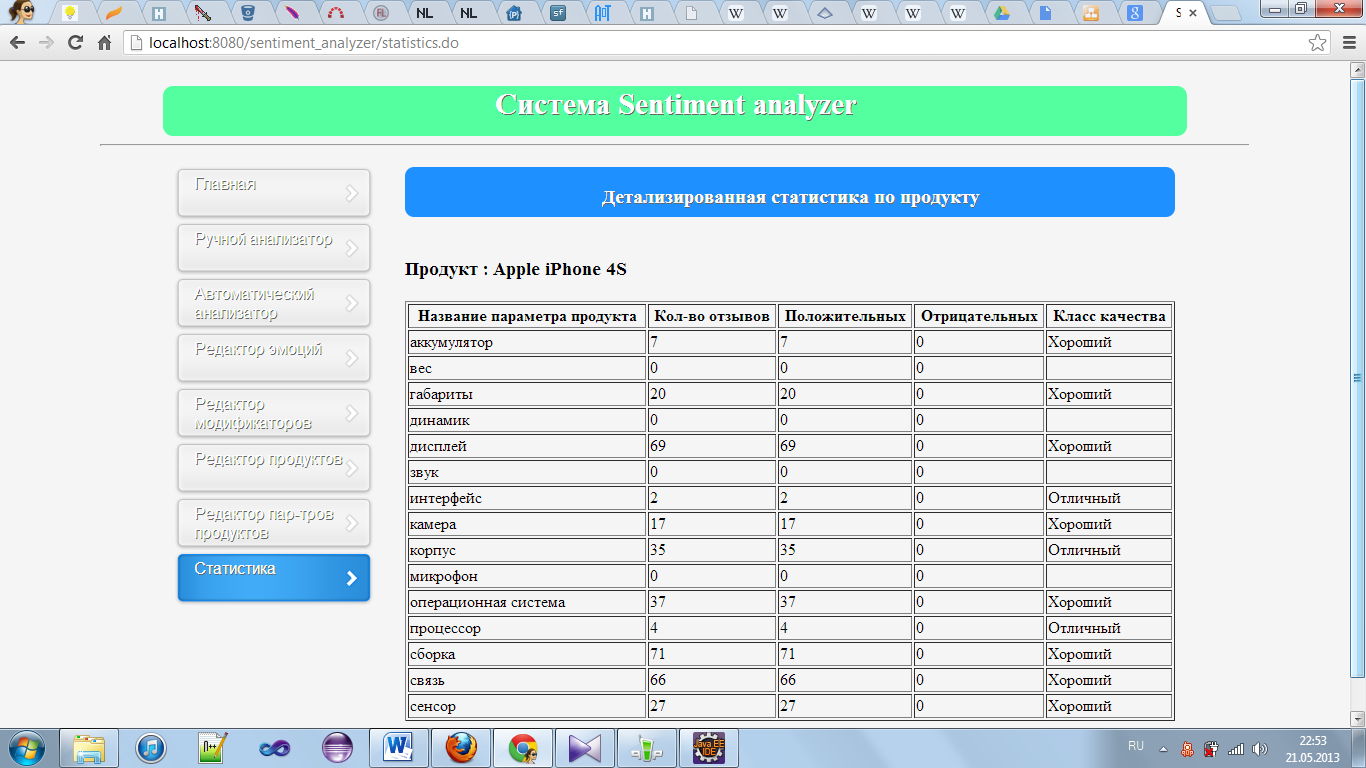


Рисунок 2.27. Статистика по параметрам продукта.

*Сообщения пользователю*

Запрос на подтверждение удаления сущностей (Рисунок 2.28.).

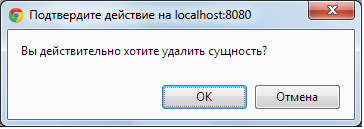


Рисунок 2.28.Удаление сущности

При успешном добавлении задачи для автоматического анализа отзывов о выбранном продукте на Яндекс маркете выводится соответствующее сообщение.

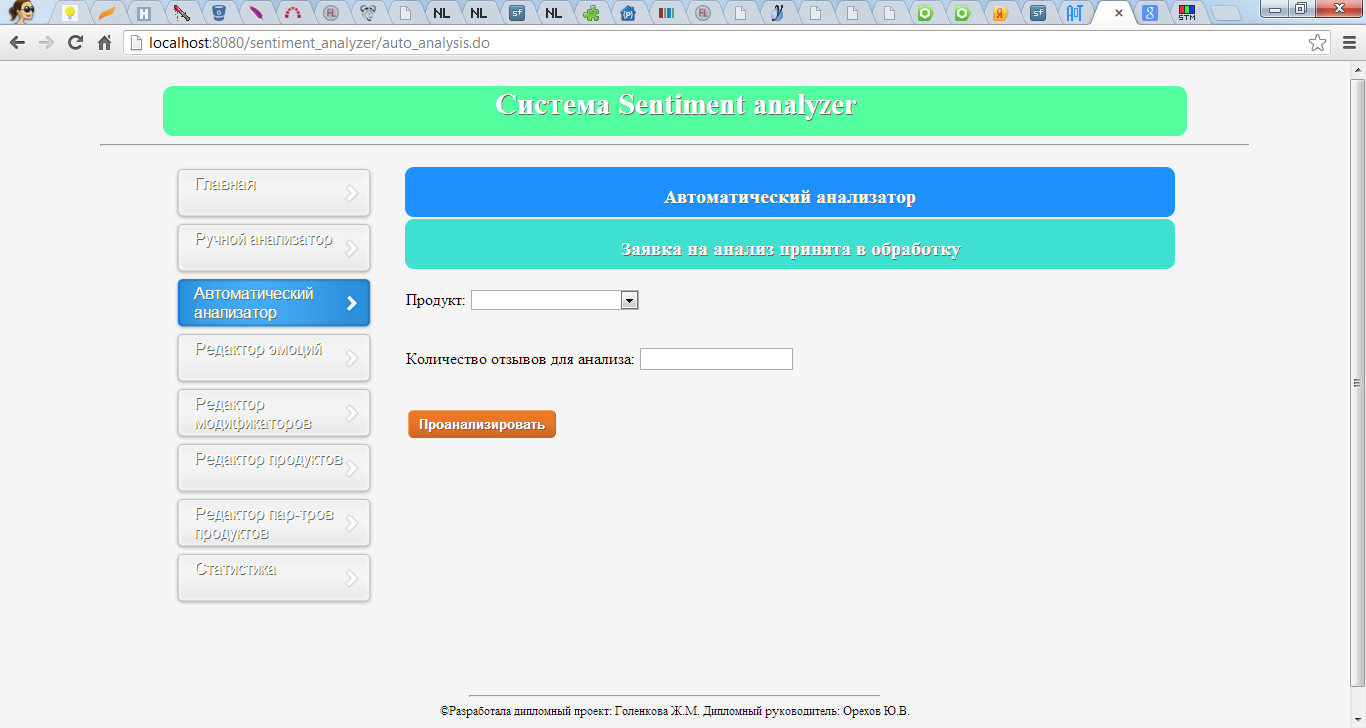


Рисунок 2.29.Сообщение об успешном добавлении задачи.

## Оценка качества программного обеспечения

Для оценки качества программного обеспечения воспользуемся ГОСТ 28195-1989 «Оценка качества программных средств», согласно которому данный программный продукт следует отнести к подклассу 509 – прочие ПС. Следовательно, выбираются показатели качества, зависящие от области применения с учетом требований области применения. Обозначения:

«+» — применяемость;

«–» — неприменяемость соответствующих показателей ПС;

«+/–» — ограниченная применяемость.

Таблица №2.1. Номенклатура показателей качества, характеризующих свойства программного средства.

|  |  |  |
| --- | --- | --- |
| Наименование групп и комплексных показателей качества | Обозначение показателя |  |
| 1. **Показатели надежности ПС** |  |  |
| 1.1. Устойчивость функционирования | Н1 | «+/–» |
| 1.2. Работоспособность | Н2 | «+» |
| 2. **Показатели сопровождения** |  |  |
| 2.1. Структурность | С1 | «+/–» |
| 2.2. Простота конструкции | С2 | «–» |
| 2.3 Наглядность | С3 | «+/–» |
| 2.4. Повторяемость | С4 | «–» |
| 3. **Показатели удобства применения** |  |  |
| 3.1. Легкость освоения | У1 | «+» |
| 3.2. Доступность эксплуатационных программных документов | У2 | «+» |
| 3.3. Удобство эксплуатации и обслуживания | У3 | «+» |
| 4. **Показатели эффективности** |  |  |
| 4.1. Уровень автоматизации | Э1 | «+/–» |
| 4.2. Временная эффективность | Э2 | «+/–» |
| 4.3. Ресурсоемкость | Э3 | «–» |
| 5.**Показатели универсальности** |  |  |
| 5.1. Гибкость | Г1 | «+/–» |
| 5.2. Мобильность | Г2 | «+» |
| 5.3. Модифицируемость | Г3 | «+/–» |
| 6.**Показатели корректности** |  |  |
| 6.1. Полнота реализации | К1 | «+» |
| 6.2. Согласованность | К2 | «+» |
| 6.3. Логическая корректность | К3 | «–» |
| 6.4. Проверенность | К4 | «+» |

1. **Фактор надежности (ФН).**

Таблица 2.2. Критерий устойчивость функционирования (УФ).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Н0101 | Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных | Экспертный  0-1 | 1 | 0,63 |
| Н0102 | Возможность обработки ошибочных ситуаций | 0-1 | 0,9 |
| Н0103 | Полнота обработки ошибочных ситуаций | 0-1 | 0,5 |
| Н0104 | Наличие тестов для проверки допустимых значений входных данных | 0-1 | 0 |
| Н0105 | Наличие системы контроля полноты входных данных | 0-1 | 1 |
| Н0106 | Наличие средств контроля корректности входных данных | 0-1 | 1 |
| Н0107 | Наличие средств контроля непротиворечивости входных данных | 0-1 | 0 |
| Н0108 | Наличие проверки параметров и адресов по диапазону их значений | 0-1 | 0,7 |
| Н0109 | Наличие обработки граничных результатов | 0-1 | 0,2 |
| Н0110 | Наличие обработки неопределенностей | 0-1 | 1 |
| Н0201 | Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств | 0-1 | 0 | 0,1 |
| Н0202 | Наличие требований к программе по восстановлению результатов при отказах процессора, ОС | 0-1 | 0 |
| Н0203 | Наличие средств восстановления процесса в случае сбоев оборудования | 0-1 | 0 |
| Н0204 | Наличие возможности разделения по времени выполнения отдельных функций программы | 0-1 | 0,5 |
| Н0205 | Наличие возможности повторного старта с точки останова | 0-1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Н0301 | Наличие централизованного управления процессами, конкурирующими из-за ресурсов | 0-1 | 0 | 0,574 |
| Н0302 | Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления | 0-1 | 1 |
| Н0303 | Наличие средств, обеспечивающих завершение процесса решения в случае помех | 0-1 | 0 |
| Н0304 | Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок или помех | 0-1 | 1 |
| Н0305 | Показатель устойчивости к искажающим воздействиям | Расчетный  ,  – число экспериментов, в которых искажающие воздействия приводили к отказу,  – число экспериментов, в которых имитировались искажающие воздействия. | P = 1-2/15=0,87 |

Вес критерия:

УФ=(Н01+Н02+Н03)/3=(0,63+0,1+0,574)/3=0,4346.

Таблица 2.3. Критерий работоспособность (Р).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Н0401 | Вероятность безотказной работы | Расчетный  *P = 1 - Q/N,*  где *Q* - число зарегистрированных отказов,  *N* - число экспериментов | P = 1 -0.2=0.8 | 0,8 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Н0501 | Оценка по среднему времени восстановления | Расчетный    где  - допустимое среднее время восстановления;  - среднее время восстановления, которое определяется по формуле:    где  - число восстановлений;  - время восстановления после i-го отказа | =(3+4+3+2+1+3+2+4+3+2)/10= 2.8 с  = 2,5 с  *Q* = = 2,5/2.8 = 0.89 | 0,93 |
| Н0502 | Оценка по продолжительности преобразования входного набора данных в выходной | Расчетный    - допустимое время преобразования *i*-го входного набора данных,  – фактическая продолжительность преобразования *i*-го входного набора данных  *N=3* (кол-во входных наборов данных). | = 0.76 с  = 1 с  = 1  = 1.1 с  = 1 с  = 1/1.1 = 0.91  = 0.86 с  = 1 с  = 1  = (1+0.91+1)/3 = 0.97 |

Вес критерия:

Р=(Н04+Н05)/2=(0,8+0,93)/2=0,865.

ФН=(УФ+Р)/2=(0,4346+0,865)/2=0,65.

1. **Фактор сопровождаемость (ФC).**

Таблица 2.4. Критерий структурность (С).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| С0601 | Использование при построении программ метода структурного программирования | 0-1 | 1 | 0,75 |
| С0602 | Соблюдение принципа разработки программ сверху вниз | 0-1 | 1 |
| С0603 | Оценка программы по числу циклов с одним входом и одним выходом | 0-1 | 0,5 |
| С0604 | Оценка программы по числу циклов | 0-1 | 0,5 |

Вес критерия:

С =(С06)/1=3/4=0,75.

Таблица 2.5. Критерий наглядность (Н).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| С0801 | Наличие комментариев ко всем машинозависимым частям программ | 0-1 | 1 | 1 |
| С0802 | Наличие комментариев к машинозависимым операторам программ | 0-1 | 1 |
| С0803 | Наличие комментариев в точках входа и выхода программы | 0-1 | 1 |
| С0901 | Соответствие комментариев принятым соглашениям | 0-1 | 1 | 0,33 |
| С0902 | Наличие комментариев-заголовков программы с указанием ее структурных и функциональных характеристик | 0-1 | 0 |
| С0903 | Оценка ясности и точности описания последовательности функционирования всех элементов программы | 0-1 | 0 |

Вес критерия:

Н = (С08+С09)/2=(1+0,33)/2=0,665.

ФС = (С+Н)/2=(0,75+0,665)/2=0,71.

1. **Фактор удобство применения (ФУП).**

Таблица 2.6. Критерий легкость освоения (ЛУ).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| У0101 | Возможность освоения программных средств документации | 0-1 | 1 | 1 |
| У0102 | Возможность освоения ПС на контрольном примере при помощи ЭВМ | 0-1 | 1 |
| У0103 | Возможность поэтапного освоения ПС | 0-1 | 1 |
| У0201 | Полнота и понятность документации для освоения | 0-1 | 1 | 1 |
| У0202 | Точность документации для освоения | 0-1 | 1 |
| У0203 | Техническое исполнение документации | 0-1 | 1 |

Вес критерия:

ЛУ = (У01+У02)/2=(1+1)/2=1

Таблица 2.7. Критерий доступность эксплуатационных программных документов (ДЭПД).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| У0301 | Наличие краткой аннотации | 0-1 | 1 | 0,82 |
| У0302 | Наличие описания решаемых задач | 0-1 | 1 |
| У0303 | Наличие описания структуры функции ПС | 0-1 | 1 |
| У0304 | Наличие описания основных функций ПС | 0-1 | 1 |
| У0306 | Наличие описания частных функций ПС | 0-1 | 0 |
| У0307 | Наличие описания алгоритмов | 0-1 | 0,5 |
| У0308 | Наличие описания межмодульных интерфейсов | 0-1 | 1 |
| У0309 | Наличие описания пользовательских интерфейсов | 0-1 | 1 |
| У0310 | Наличие описания входных и выходных данных | 0-1 | 1 |
| У0311 | Наличие описания диагностических сообщений | 0-1 | 1 |
| У0312 | Наличие описания основных характеристик ПС | 0-1 | 1 |
| У0314 | Наличие описания программной среды функционирования ПС | 0-1 | 1 |
| У0315 | Достаточность документации для ввода ПС в эксплуатацию | 0-1 | 1 |
| У0316 | Наличие информации технологии переноса для мобильных программ | 0-1 | 0 |
| У0401 | Соответствие оглавления содержанию документации | 0-1 | 1 | 1 |
| У0402 | Оценка оформления документации | 0-1 | 1 |
| У0403 | Грамматическая правильность изложения документации | 0-1 | 1 |
| У0404 | Отсутствие противоречий | 0-1 | 1 |
| У0405 | Отсутствие неправильных ссылок | 0-1 | 1 |
| У0406 | Ясность формулировок и описаний | 0-1 | 1 |
| У0407 | Отсутствие неоднозначных формулировок и описаний | 0-1 | 1 |
| У0408 | Правильность использования терминов | 0-1 | 1 |
| У0409 | Краткость, отсутствие лишней детализации | 0-1 | 1 |
| У0410 | Единство формулировок | 0-1 | 1 |
| У0411 | Единство обозначений | 0-1 | 1 |
| У0412 | Отсутствие ненужных повторений | 0-1 | 1 |
| У0413 | Наличие нужных объяснений | 0-1 | 1 |
| У0501 | Оценка стиля изложения | 0-1 | 0,5 | 0,58 |
| У0502 | Дидактическая разделенность | 0-1 | 0 |
| У0503 | Формальная разделенность | 0-1 | 1 |
| У0504 | Ясность логической структуры | 0-1 | 1 |
| У0505 | Соблюдение стандартов и правил изложения в документации | 0-1 | 1 |
| У0506 | Оценка по числу ссылок вперед в тексте документа | 0-1 | 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| У0601 | Наличие оглавления | 0-1 | 1 | 0,78 |
| У0602 | Наличие предметного указателя | 0-1 | 0 |
| У0603 | Наличие перекрестных ссылок | 0-1 | 0 |
| У0604 | Наличие всех требуемых разделов | 0-1 | 1 |
| У0605 | Соблюдение непрерывности нумерации страниц документов | 0-1 | 1 |
| У0606 | Отсутствие незаконченных разделов, абзацев, предложений | 0-1 | 1 |
| У0607 | Наличие всех рисунков, чертежей, формул, таблиц | 0-1 | 1 |
| У0608 | Наличие всех строк и примечаний | 0-1 | 1 |
| У0609 | Логический порядок частей внутри главы | 0-1 | 1 |
| У0701 | Наличие полного перечня документации | 0-1 | 1 | 1 |

Вес критерия:

ДЭПД = (У03+У04+У05+У06+У07)/5 = (0,82+1+0,58+0,78+1)/5 = 0,84.

Таблица 2.8. Критерий удобство эксплуатации и обслуживания (УЭО).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| У0801 | Уровень языка общения пользователя с программой | 0-1 | 1 | 0,8 |
| У0802 | Легкость и быстрота загрузки и запуска программы | 0-1 | 1 |
| У0803 | Легкость и быстрота завершения программы | 0-1 | 1 |
| У0804 | Возможность распечатки содержимого программы | 0-1 | 1 |
| У0805 | Возможность приостанова и повторного запуска без потерь информации | 0-1 | 0 |
| У0901 | Соответствие меню требованиям пользователя | 0-1 | 1 | 0,5 |
| У0902 | Возможность перехода вверх и вниз по многоуровневому меню | 0-1 | 0 |
| У1001 | Возможность управления подробностью получаемых выходных данных | 0-1 | 1 | 1 |
| У1002 | Достаточность полученной информации для продолжения работы | 0-1 | 1 |
| У1101 | Обеспечение удобства ввода данных | 0-1 | 1 | 1 |
| У1102 | Легкость восприятия | 0-1 | 1 |
| У1201 | Обеспечение программой выполнения предусмотренных рабочих процедур | 0-1 | 1 | 1 |
| У1202 | Достаточность информации, выдаваемой программой для составления дополнительных процедур | 0-1 | 1 |

Вес критерия:

УЭО = (У08+У09+У10+У11+У12)/5 = (0,8+0,5+1+1+1)/5 = 0,86.

ФУП = (ЛУ+ДЭПД+УЭО)/3 = (1+0,84+0,86)/3 = 0,9.

1. **Фактор эффективность (ФЭ).**

Таблица 2.9. Критерий уровень автоматизации (УА).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Э0101 | Проблемно-ориентированные функции | 0-1 | 1 | 0,9 |
| Э0102 | Машинно-ориентированные функции | 0-1 | 1 |
| Э0103 | Функции ведения и управления | 0-1 | 1 |
| Э0104 | Функции ввода/вывода | 0-1 | 1 |
| Э0105 | Функции защиты и проверки данных | 0-1 | 1 |
| Э0106 | Функции защиты от несанкционированного доступа | 0-1 | 1 |
| Э0107 | Функции контроля доступа | 0-1 | 1 |
| Э0108 | Функции защиты от внесения изменений | 0-1 | 1 |
| Э0109 | Наличие соответствующих границ функциональных областей | 0-1 | 1 |
| Э0110 | Число знаков после запятой в результатах вычислений | 0-1 | 0 |
| Э0201 | Время выполнения программ | 0-1 | 0,8 | 0,6 |
| Э0202 | Время реакции и ответов | 0-1 | 0,9 |
| Э0203 | Время подготовки | 0-1 | 0,5 |
| Э0205 | Затраты времени на защиту данных | 0-1 | 0 |
| Э0206 | Время компиляции | 0-1 | 0,8 |

Вес критерия:

УА = (Э01+Э02)/2 = (0,9+0,6)/2 = 0,75.

Таблица 2.10. Критерий временная эффективность (ВЭ).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Э0301 | Требуемый объем внутренней памяти | 0-1 | 0.8 | 0,575 |
| Э0302 | Требуемый объем внешней памяти | 0-1 | 0.7 |
| Э0303 | Требуемые периферийные устройства | 0-1 | 0 |
| Э0304 | Требуемое базовое программное обеспечение | 0-1 | 0.8 |

Вес критерия:

ВЭ = (Э03)/1 = 0,575.

ФЭ = (УА +ВЭ)/2 = (0,75+0,575)/2 = 0,66.

1. **Фактор универсальность (ФУн).**

Таблица 2.11. Критерий гибкость (Г).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Г0101 | Оценка числа потенциальных пользователей | 0-1 | 1 | 0,9 |
| Г0102 | Оценка числа функций ПС | 0-1 | 0,8 |
| Г0103 | Насколько набор функций удовлетворяет требованиям пользователя | 0-1 | 0,9 |
| Г0104 | Насколько возможности программ охватывают область решаемых пользователем задач | 0-1 | 0,8 |
| Г0105 | Возможность настройки формата выходных данных для конкретных пользователей | 0-1 | 1 |
| Г0201 | Наличие схемы иерархии модулей программы | 0-1 | 1 | 0,61 |
| Г0202 | Оценка независимости модулей | 0-1 | 0,5 |
| Г0203 | Оценка числа уникальных элементов/реквизитов | 0-1 | 0,7 |
| Г0204 | Используется ли в текущем вызове модуля информация, полученная в предыдущем вызове | 0-1 | 1 |
| Г0205 | Оценка организации точек входа и выхода модуля | 0-1 | 0,5 |
| Г0206 | Наличие описания атрибутов модуля | 0-1 | 0 |
| Г0301 | Оценка программ по числу переходов и точек ветвления | 0-1 | 0,5 | 0,5 |
| Г0401 | Использование метода пошагового уточнения | 0-1 | 0,75 | 0,83 |
| Г0402 | Наличие описания структуры программ | 0-1 | 0,8 |
| Г0403 | Наличие описания связей между элементами структуры программы | 0-1 | 0,9 |
| Г0404 | Наличие в программе повторного выполнения функций (подпрограмм) | 0-1 | 0,9 |
| Г0501 | Использование стандартных протоколов связи | 0-1 | 1 | 1 |

Вес критерия:

Г = (Г01+Г02+Г03+Г04+Г05)/5 =(0,9+0,61+0,5+0,83+1)/5= 0,768.

Таблица 2.12. Критерий мобильность (Моб).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Г0701 | Оценка зависимости программ от емкости оперативной памяти ЭВМ | 0-1 | 0,8 | 0,525 |
| Г0702 | Оценка зависимости временных характеристик программы от скорости вычисления ЭВМ | 0-1 | 0,9 |
| Г0703 | Оценка зависимости функционирования программы от числа внешних запоминающих устройств и их общей емкости | 0-1 | 0,3 |
| Г0704 | Оценка зависимости функционирования программы от специальных устройств ввода-вывода | 0-1 | 0,1 |
| Г0801 | Применение специальных языков программирования | 0-1 | 1 | 0,76 |
| Г0802 | Оценка зависимости программы от программ операционной системы | 0-1 | 0,5 |
| Г0803 | Зависимость от других программных средств | 0-1 | 0,8 |
| Г0901 | Оценка локализации непереносимой части программы | 0-1 | 0,1 | 0,1 |

Вес критерия:

Моб = (Г07+Г08+Г09)/3 =(0,525+0,76+0,1)/3= 0,462.

Таблица 2.13. Критерий модифицируемость (Мод).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| Г1001 | Оценка использования отрицательных или булевых выражений | 0-1 | 0,6 | 0,57 |
| Г1002 | Оценка программы по использованию условных переходов | 0-1 | 0,8 |
| Г1003 | Оценка программы по использованию безусловных переходов | 0-1 | 0 |
| Г1004 | Оформление процедур входа и выхода из циклов | 0-1 | 0,8 |
| Г1005 | Ограничения на модификацию переменной индексации в цикле | 0-1 | 0,4 |
| Г1006 | Оценка модулей по направлению потока управления | 0-1 | 0,4 |
| Г1007 | Оценка программы по использованию локальных переменных | 0-1 | 1 |
| Г1101 | Оценка программы по числу комментариев | 0-1 | 0,9 | 0,9 |
| Г1201 | Наличие заголовка в программе | 0-1 | 1 | 0,575 |
| Г1202 | Комментарии к точкам ветвлений | 0-1 | 0,6 |
| Г1203 | Комментарии к машинозависимым частям программы | 0-1 | 1 |
| Г1204 | Комментарии к машинозависимым операторам программы | 0-1 | 1 |
| Г1205 | Комментарии к операторам объявления переменных | 0-1 | 1 |
| Г1206 | Оценка семантики операторов | 0-1 | 0 |
| Г1207 | Наличие соглашений по форме представления комментариев | 0-1 | 0 |
| Г1208 | Наличие общих комментариев к программам | 0-1 | 0 |
| Г1301 | Использование языков высокого уровня | 0-1 | 1 | 0,73 |
| Г1302 | Семантика имен используемых переменных | 0-1 | 0,7 |
| Г1303 | Использование отступов, сдвигов и пропусков при формировании текста | 0-1 | 1 |
| Г1304 | Размещение операторов по строкам | 0-1 | 0,2 |
| Г1401 | Передача информации для управления по параметрам | 0-1 | 0,2 | 0,68 |
| Г1402 | Параметрическая передача входных данных | 0-1 | 0,3 |
| Г1403 | Наличие проверки результатов работы между модулями | 0-1 | 1 |
| Г1404 | Наличие проверки правильности данных, получаемых модулями от вызываемого модуля | 0-1 | 0,9 |
| Г1405 | Использование общих областей памяти | 0-1 | 1 |

Вес критерия:

Мд = (Г10+Г11+Г12+Г13+Г14)/5 =(0,57+0,9+0,56+0,73+0,68)/5= 0,69.

ФУн = (Г+Моб+Мд)/3 = (0,768+0,462+0,69)/3 = 0,64.

1. **Фактор корректность (Фкт).**

Таблица 2.14. Критерий полнота реализации (ПРл).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| К0101 | Наличие всех необходимых документов для понимания и использования ПС | 0-1 | 1 | 0,76 |
| К0102 | Наличие описания и схемы иерархии модулей программы | 0-1 | 0,7 |
| К0103 | Наличие описания основных функций | 0-1 | 1 |
| К0104 | Наличие описания частных функций | 0-1 | 1 |
| К0105 | Наличие описания данных | 0-1 | 1 |
| К0106 | Наличие описаний алгоритмов | 0-1 | 1 |
| К0107 | Наличие описаний интерфейсов между модулями | 0-1 | 1 |
| К0108 | Наличие описаний интерфейсов с пользователями | 0-1 | 1 |
| К0109 | Наличие описания используемых числовых методов | 0-1 | 0 |
| К0110 | Указаны ли все численные методы | 0-1 | 0 |
| К0111 | Наличие описаний всех параметров | 0-1 | 1 |
| К0112 | Наличие описания методов настройки системы | 0-1 | 1 |
| К0113 | Наличие описаний всех диагностических сообщений | 0-1 | 1 |
| К0114 | Наличие описания способов проверки работоспособности | 0-1 | 0 |
| К0201 | Реализация всех исходных модулей | 0-1 | 0,9 | 0,63 |
| К0202 | Реализация всех основных функций | 0-1 | 0,9 |
| К0203 | Реализация всех частных функций | 0-1 | 1 |
| К0204 | Реализация всех алгоритмов | 0-1 | 1 |
| К0205 | Реализация всех взаимосвязей в системе | 0-1 | 0,8 |
| К0206 | Реализация всех интерфейсов между модулями | 0-1 | 0,5 |
| К0207 | Реализация возможности настройки системы | 0-1 | 0 |
| К0208 | Реализация диагностики всех граничных и аварийных ситуаций | 0-1 | 0 |
| К0209 | Наличия определения всех данных (индексы, массивы и проч.) | 0-1 | 0,2 |
| К0210 | Наличие интерфейсов с пользователем | 0-1 | 1 |

Вес критерия:

ПРл = (К01+К02)/2 =(0,76+0,63)/2 = 0,7.

Таблица 2.15. Критерий согласованность (Сг).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| К0301 | Отсутствие противоречий в описании частных функций | 0-1 | 1 | 1 |
| К0302 | Отсутствие противоречий в описании основных функций в разных документах | 0-1 | 1 |
| К0303 | Отсутствие противоречий в описании алгоритмов | 0-1 | 1 |
| К0304 | Отсутствие противоречий в описании взаимосвязей в системе | 0-1 | 1 |
| К0305 | Отсутствие противоречий в описании интерфейсов между модулями | 0-1 | 1 |
| К0306 | Отсутствие противоречий в описании интерфейсов с пользователем | 0-1 | 1 |
| К0307 | Отсутствие противоречий в описании настройки системы | 0-1 | 1 |
| К0309 | Отсутствие противоречий в описании иерархической структуры сообщений | 0-1 | 1 |
| К0310 | Отсутствие противоречий в описании диагностических сообщений | 0-1 | 1 |
| К0311 | Отсутствие противоречий в описании данных | 0-1 | 1 |
| К0401 | Отсутствие противоречий в выполнении основных функций | 0-1 | 1 | 0,98 |
| К0402 | Отсутствие противоречий в выполнении частных функций | 0-1 | 1 |
| К0403 | 0тсутствие противоречий в выполнении алгоритмов | 0-1 | 1 |
| К0404 | Правильность взаимосвязей | 0-1 | 0,9 |
| К0405 | Правильность реализации интерфейса между модулями | 0-1 | 0,9 |
| К0406 | Правильность реализации интерфейса с пользователем | 0-1 | 1 |
| К0407 | Отсутствие противоречий в настройке системы | 0-1 | 1 |
| К0408 | Отсутствие противоречий в диагностике системы | 0-1 | 1 |
| К0409 | Отсутствие противоречий в общих переменных | 0-1 | 1 |
| К0501 | Единообразие способов вызова модулей | 0-1 | 1 | 0,88 |
| К0502 | Единообразие процедур возврата управления из модулей | 0-1 | 1 |
| К0503 | Единообразие способов сохранения информации для возврата | 0-1 | 1 |
| К0504 | Единообразие способов восстановления информации для возврата | 0-1 | 0,8 |
| К0505 | Единообразие организации списков передаваемых параметров | 0-1 | 0,6 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| К0601 | Единообразие именования каждой переменной и константы | 0-1 | 0,5 | 0,58 |
| К0602 | Все ли одинаковые константы встречаются во всех программах под одинаковыми именами | 0-1 | 0,9 |
| К0603 | Единообразие определения внешних данных во всех программах | 0-1 | 0,9 |
| К0604 | Используются ли разные идентификаторы для разных переменных | 0-1 | 0,9 |
| К0605 | Все ли общие переменные объявлены как общие переменные | 0-1 | 0,2 |
| К0606 | Наличие определений одинаковых атрибутов | 0-1 | 0,1 |
| К0701 | Комплектность документации в соответствии со стандартами | 0-1 | 0,9 | 0,97 |
| К0702 | Правильное оформление частей документов | 0-1 | 1 |
| К0703 | Правильное оформление титульных и заглавных листов документов | 0-1 | 1 |
| К0704 | Наличие в документах всех разделов в соответствии со стандартами | 0-1 | 1 |
| К0705 | Деление документов на структурные элементы | 0-1 | 1 |
| К0706 | Полнота содержания разделов в соответствии со стандартами | 0-1 | 0,9 |
| К0801 | Соответствие организации и вычислительного процесса эксплуатационной документации | 0-1 | 1 | 0,97 |
| К0802 | Правильность заданий на выполнение программы, правильность написания управляющих операторов (отсутствие ошибок) | 0-1 | 0,9 |
| К0803 | Отсутствие ошибок в описании действий пользователя | 0-1 | 0,9 |
| К0804 | Отсутствие ошибок в описании запуска | 0-1 | 1 |
| К0805 | Отсутствие ошибок в описании генерации | 0-1 | 1 |
| К0806 | Отсутствие ошибок в описании настройки | 0-1 | 1 |

Вес критерия:

Сг=(К03+К04+К05+К06+К07+К08)/6=(1+0,98+0,88+0,58+0,97+0,97)/6=0,896.

Таблица 2.16. Критерий проверенность (Пт).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код  ОЭ | Наименование оценочного элемента | Метод  оценки | Оценка | Вес  метрики |
| К1001 | Наличие требований к тестированию программ | 0-1 | 0,6 | 0,68 |
| К1002 | Достаточность требований к тестированию программ | 0-1 | 0,5 |
| К1003 | Отношение числа модулей, отработавших в процессе тестирования и отладки (*Qтм*) к общему числу модулей (*Qoм*) | Расчетный  *Qтм /Qом*  *Qтм* =8  *Qом* =9 | 8/9 = 0.88 |
| К1004 | Отношение числа логических блоков, отработавших в процессе тестирования и отладки (*Qтб*) к общему числу логических блоков (*Qoб*) | Расчетный  *Qтб* =27  *Qоб* =35 | 27/35 = 0.77 |

Вес критерия:

Пт = 0,68.

ФКт = (ПРл+Сг+Пт)/3 =(0,7+0,896+0,68)/3= 0,758.

**ВЫВОД.**

В результате проведенного анализа получили следующие оценки факторов качества программного средства:

|  |  |
| --- | --- |
| Фактор | Оценка |
| Надежность (ФН) | 0,65 |
| Сопровождение (ФС) | 0,71 |
| Удобство применения (ФУП) | 0,9 |
| Эффективность (ФЭ) | 0,66 |
| Универсальность (ФУн) | 0,64 |
| Корректность (ФКт) | 0,758 |

Для улучшения оценок необходимо сделать:

1. Надежность. Реализовать обработку критических мест в программе. Реализовать проверку неопределенностей. Так же можно дополнить возможностью повторного старта с точки останова.
2. Эффективность. Этот фактор в большей степени зависит от программно-технической оснащенности сервера, нежели чем от самого ПС. И поэтому оценочные показатели были установлены усреднено в зависимости от технического оснащения сервера. Т.е. требования, которые предъявляет ПС к объемам внутренней и внешней памяти на сегодняшний день являются актуальными. А вот требуемое базовое программное обеспечение может просто не оказаться у отдельно взятого пользователя. А это может привести к полной неработоспособности ПС
3. Универсальность. Описать связи между элементами структуры программы.

# ЗАКЛЮЧЕНИЕ

В настоящее время пользователи различных товаров активно делятся своим мнением о товаре в Интернете, поэтому производителям важно узнавать актуальное мнение об их товаре. На сегодняшний день существует несколько программных продуктов, анализирующих тональность текстовых отзывов по определенным предметным областям. Анализ данных программных продуктов показал, что необходимо разработать собственную математическую модель и программный продукт для решения поставленной задачи.

В представленной работе разработан модифицированный метод на основе составления словарей, дополненный графематическим анализом и алгоритмом нечеткого поиска по словарю. Также была реализована интеграция с интернет ресурсом Яндекс маркет. Был выбран подход к разработке программного обеспечения, изучены методы взаимодействия JAVA, HTML и БД MySQL.

В ходе выполнения выпускной дипломной работы:

Была достигнута цель: повышена эффективность автоматизации оценки качества товара на основе разработки математического и программного обеспечения по распознаванию тональности текстовых отзывов из сети Интернет.

Были решены поставленные задачи:

1. Произведен обзор существующих методов автоматического анализа тональности текстовых отзывов.
2. Произведено исследование особенностей текстовых сообщений в сети Интернет в контексте разработки метода анализа тональности текстовых отзывов.
3. Произведен обзор и анализ алгоритмов нечеткого поиска.
4. Задача оценки качества товара поставлена как задача классификации в терминах теории распознавания образов.
5. Разработан метод автоматического анализа тональности текстовых отзывов из сети Интернет. За основу был взят метод на основе словарей и дополнен графематическим анализом и алгоритмом расширения выборки.
6. Разработан метод определения класса качества товара и параметров товара.
7. Разработан программный продукт. Программный продукт внедрен в ООО «СофтПрофи».
8. Проанализированы результаты, точность оценки качества товара составляет 71,9%.

# СПИСОК ЛИТЕРАТУРЫ

1. Пазельская А., Соловьев А. Метод определения эмоций в текстах на русском языке: труды международной конференции "Диалог, 2011". P.510– 522.
2. Веб-сервис Twitter Sentiment <http://www.sentiment140.com>
3. Mitchell T. Machine Learning. — McGraw-Hill Science/Engineering/Math, 1997. ISBN 0-07-042807-7.
4. Веб-сервис I-Teco <http://x-file.su/tm/Default.aspx>
5. Веб-сервис Sentiment Analysis with Python NLTK Text Classification <http://text-processing.com/demo/sentiment/>
6. Наивный-байесовский-классификатор <http://ru.wikipedia.org/wiki/Наивный_байесовский_классификатор>
7. Liu H. MontyLingua: An end-to-end natural language processor with common sense, 2004. <http://web.media.mit.edu/~hugo/montylingua>
8. Joachims T. Making large-scale SVM learning practical // In B. Schölkopf, C. J. C. Burges, & A. J. Smola (Eds.), Advances in kernel methods: support vector learning, 1999. The MIT Press.
9. Turney P. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews // Proceedings of ACL-02, 40th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2002, pp. 417–424.
10. König A. C. & Brill, E. Reducing the human overhead in text categorization // In Proceedings of the 12th ACM SIGKDD conference on knowledge discovery and data mining, August 20–23, 2006, pp. 598–603.
11. Расстояние Хэмминга <http://ru.wikipedia.org/wiki/Расстояние_Хэмминга>
12. Расстояние Левенштейна <http://ru.wikipedia.org/wiki/Расстояние_Левенштейна>
13. Расстояние Дамерау-Левенштейна. <http://en.wikipedia.org/wiki/Damerau–Levenshtein_distance>
14. Метод динамического программирования Вагнера и Фишера <http://algolist.manual.ru/search/lcs/vagner.php>
15. Метод N-грамм. <http://www.cs.helsinki.fi/u/ukkonen/TCS92.pdf>
16. Open-source library Snowball stemmer <http://snowball.tartarus.org/>
17. Open-source library FreeLing <http://devel.cpl.upc.edu/freeling>

**Нормативно – технические документы**

1. <http://rgost.ru/> - ГОСТ 19.404, ГОСТ Р 50779.10-2000, ГОСТ Р 50779.11-2000, ГОСТ 16504-81, ГОСТ 24026-80, ГОСТ 19.701-90, ГОСТ 19.781-90, ГОСТ 28195–89.
2. <http://gosthelp.ru/> - ГОСТ Р ИСО/МЭК 9126-93, ГОСТ 34.320-96, ГОСТ Р 50.1.028.–2001, ГОСТ Р ИСО/МЭК 12207-99, ISO/IEC 19501

# ПРИЛОЖЕНИЕ 1

# ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

**Расчет цены предложения и условий окупаемости**

**разработанного программного продукта**

1. Расчет себестоимости машиночаса эксплуатации

вычислительной и оргтехники.

Себестоимость машиночаса (*См-ч*) эксплуатации вычислительной и оргтехники может быть рассчитана следующим образом:

 (1)

где – суммарные годовые затраты, связанные с эксплуатацией и обслуживанием вычислительной и оргтехники (руб./год);

*КГ* – коэффициент готовности вычислительной и оргтехники (*КГ* = 0,95);

*FП* – годовой эффективный (полезный) фонд времени работы единицы оборудования вычислительной и оргтехники (ч./год).

Годовой эффективный (полезный) фонд времени работы единицы оборудования может быть рассчитан следующим образом:

 (2)

где  – коэффициент, учитывающий время затрачиваемое на ремонт, обслуживание, настройку и пр. (= 0,14);

*FН*  – годовой номинальный фонд времени работы оборудования (в 2013-м году, *FН =* 1986 ч/год, что соответствует номинальному фонду времени при односменном режиме работы).



Для расчета суммарных годовых затрат, связанных с эксплуатацией и обслуживанием вычислительной и оргтехники необходимо определить балансовую стоимость условного комплекта вычислительной и оргтехники, с помощью которого выполнялась разработка программного продукта.

Таблица 1 – Состав и стоимость объектов входящих в условный комплект вычислительной и оргтехники.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование объекта | Количество  (единиц) | Цена за единицу  (руб.) |
| 1. | Системный блок DNS Prestige [0158676] | 1 | 24990 |
| 2. | LCD монитор Dell UltraSharp U2412M | 1 | 13490 |
| 3 | Клавиатура Oklick 120M black PS/2 | 1 | 290 |
| 4 | Оптическая мышь A4tech X5-50D-2 | 1 | 280 |
| 5 | Принтер HP LaserJet Enterprise P3015dn | 1 | 21890 |
|  | ИТОГО: |  | 60940 |

Таким образом, балансовая стоимость условного комплекта (*Сб*) может быть определена следующим образом:

 (3)

где *Цi* – цена единицы оборудования, входящего в условный комплект вычислительной и оргтехники (руб./ед.);

*Ki* – количество единиц оборудования, входящих в комплект (ед.);

*РД* – дополнительные расходы на доставку, установку, первоначальную настройку и пр. (руб.)

Дополнительные расходы составляют 10% от цены оборудования комплекта вычислительной и оргтехники. Таким образом, *РД =* 0,1.





В перечень суммарных годовых затрат, связанных с эксплуатацией и обслуживанием вычислительной и оргтехники входят следующие.

1). Затраты на материал (*ЗМ*), которые по нормативам составляют 2% от балансовой стоимости оборудования. Таким образом:

 (4)



2). Основная и дополнительная заработная плата (*ЗЗ/П*) персонала обслуживающего вычислительную и оргтехнику.

Она может быть рассчитана из следующего выражения:

 (5)

где *ОК* – месячный оклад одного работника обслуживающего вычислительную и оргтехнику (4500руб./мес.);

*КД* – коэффициент дополнительной заработной платы (*КД* = 0,1);

*КР* – районный коэффициент(*КР* = 0,15);

*Nобсл.* – количество обслуживаемых персоналом комплектов вычислительной и оргтехники (ед.). (*Nобсл.*= 9)



3). Начисления на заработную плату (*НП*) персонала обслуживающего вычислительную и оргтехнику.

В рассматриваемом случае начисления на заработную плату состоят из взносов во внебюджетные фонды и начислений по страховому налогу от несчастного случая на производстве.

Минимально необходимые размеры страховых взносов во внебюджетные фонды законодательно устанавливаются в процентом отношении от всех затрат, произведенных в качестве оплаты труда работодателем работнику.

Таблица 2 – Минимально необходимые размеры взносов во внебюджетные фонды, законодательно установленные с 01.01.2012 г.

|  |  |  |
| --- | --- | --- |
| № | Наименование внебюджетного фонда | % начисления |
| 1. | Пенсионный фонд РФ | 22,0% |
| 2. | Фонд социального страхования РФ | 2,90% |
| 3. | Федеральный фонд обязательного медицинского  Страхования | 5,10% |
| 4. | Территориальный (региональный) фонд обязательного медицинского страхования | 0% |
|  | ИТОГО | 30% |

В связи с тем, что персонал обслуживающий вычислительную и оргтехнику относится к категории служащих, страховой налог от несчастного случая на производстве будет составлять 0,2%.

Таким образом, общий размер начислений на заработную плату персонала обслуживающего вычислительную и оргтехнику можно рассчитать как:

 (6)



4). Амортизация вычислительной и оргтехники (*АТ* ).

Годовые амортизационные отчисления (руб./год), входящие в себестоимость машиночаса эксплуатации вычислительной и оргтехники могут быть рассчитаны следующим образом:

 (7)

где  – годовая норма амортизационных отчислений вычислительной и оргтехники. (= 18%)



5). Амортизация программных продуктов (*АП*).

 (8)

где – годовая норма амортизационных отчислений программных продуктов. (= 64%);

*СП* – стоимость установленных основных программных продуктов.

Для определения размера амортизации установленных в условном комплекте вычислительной и оргтехники программных продуктов необходимо рассчитать их общую стоимость (*СП*). Состав и стоимость основных установленных программных продуктов представлен в таблице 3.

Таблица 3 – Состав и стоимость программных продуктов установленных в условном комплекте вычислительной и оргтехники.

|  |  |  |
| --- | --- | --- |
| № | Наименование программного продукта | Стоимость с установкой  (руб.) |
| 1. | Microsoft Windows 8 Professional 64-bit Russian 1pk DSP OEI DVD (ОЕМ) | 5990 |
| 2. | Антивирус Kaspersky CRYSTAL | 1450 |
| 3. | Microsoft Office Home and Business 2010 Rus | 8850 |
|  | ИТОГО: | *СП =16290* |

Основные расходы на содержание и эксплуатацию оборудования, составляющего комплект вычислительной и оргтехники, состоят из затрат на ремонт и обслуживание (*ЗР*) и затрат на электроэнергию (*ЗЭ*).



6). Затраты на ремонт и обслуживание вычислительной и оргтехники по нормативам составляют 3% от балансовой стоимости и могут быть определены как:

 (9)



7). Годовые затраты на электроэнергию, потребляемую оборудованием условного комплекта, могут быть рассчитаны следующим образом:

 (10)

где  – суммарная мощность, потребляемая всем оборудованием комплекта вычислительной и оргтехники;

*Цквтч* – цена одного киловатчаса электроэнергии (*Цквт.ч* = 2,02 руб./квт.-ч).



8). Так как в дисплейном классе используется один тип вычислительной техники и количество обслуживающего технику персонала не превышает трёх человек, то для определения себестоимости машиночаса рассчитываются, так называемые, прочие расходы (*ЗПр*), которые по нормативам составляют 75% от балансовой стоимости комплекта вычислительной и оргтехники. Таким образом, *ЗПр =* 0,75 *Сб*



Суммарные годовые затраты, связанные с эксплуатацией и обслуживанием комплекта вычислительной и оргтехники (руб./год), определяются как:

 (11)



Тогда, воспользовавшись формулой (1), можем определить себестоимость машиночаса эксплуатации вычислительной и оргтехники (*См-ч*).



1. Расчет цены предложения и тиражирования минимального

количества копий разработанного программного продукта

Для дальнейшего проведения расчетов необходимо иметь следующие исходные данные:

*ТОб* – общее время работы над программным продуктом в месяцах;

*ТМ* – время работы над программным продуктом с использованием вычислительной и оргтехники в месяцах (машинное время).

*ТОб*=4,9 мес.

*ТМ* =3,9 мес.

Общие затраты на разработку программного продукта (*ЗПП*) будут состоять из следующих основных статей расходов.

1). Фонд оплаты труда разработчиков программного продукта (*ФОТ*).

 (12)

где – суммарный месячный оклад разработчиков программного продукта (15000руб./мес.).

В связи с тем, что в данном случае время работы над программным продуктом не превышает года, *КД* = 0,15.



2). Начисления на фонд оплаты труда (*НФ*) разработчиков программного продукта. В рассматриваемом случае начисления на фонд оплаты труда состоят из взносов во внебюджетные фонды (смотри таблицу 2) и начислений по страховому налогу от несчастного случая на производстве, составляющего для разработчиков программного продукта 2% от суммы *ФОТ*.

 (13)



3). Расходы, связанные с использованием вычислительной и оргтехники (*РТ* ) могут быть рассчитаны как:

 (14)

где *КИ* – коэффициент использования оборудования, *КИ* = 0,9.

*n* – количество используемых для разработки программного продукта комплектов вычислительно и оргтехники (*n* =1ед.).



4). Расходы на специальные программные продукты (*РСП* ), необходимые для разработки. Эти расходы определяются по смете, представление о затратах которой дает таблица 4.

Таблица 4 – Состав и стоимость специальных программных продуктов, необходимых для разработки.

|  |  |  |
| --- | --- | --- |
| № | Наименование программного продукта | Стоимость с установкой  (руб.) |
| 1. | Eclipse Juno | 0 |
| 2. | MySQL Server 5.5 | 0 |
| 3 | JDK 7 | 0 |
| 4 | MySQL Workbench 5.2 | 0 |
|  | ИТОГО: | *РСП =0* |

5). Расходы на хозяйственно-операционные нужды (*РХ* ).

Эти расходы включают затраты на бумагу, магнитные носители, писчие принадлежности и т.п. Они определяются по смете, представление о которой дает таблица 5.

Таблица 5. – Состав и стоимость хозяйственно-операционных расходов.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Предмет расходов | Количество | Цена за единицу  (руб.) |
| 1. | Бумага офисная «Снегурочка», А4 | 1 | 162 |
| 2. | Дисковый накопитель Transend | 1 | 756 |
| 3. | Ручка | 2 | 28 |
| 4. | Карандаш | 1 | 14 |
| 5. | Ластик | 1 | 8 |
|  | ИТОГО: |  | *РХ =*968 |

6). Накладные расходы (*РН* ).

Накладные расходы рассчитываются по смете или по нормативам от фонда оплаты труда (*ФОТ*). В последнем случае они составляют 50% от размера *ФОТ*. Таким образом, *РН* = 0,5 *ФОТ*.



Общие затраты на разработку программного продукта (*ЗПП*) можно рассчитать как:

 (15)



Расчет установочной прибыли (*ПрУ*) производится на основе нормативной рентабельности разработки программного продукта – *RН*.

 (16)

Численной значение нормативной рентабельности разработки программного продукта *RН* = 20%.



Расчет величины налога на добавленную стоимость (*НДС*).

Величина налога на добавленную стоимость может быть рассчитана из следующего выражения:

 (17)

где *ННДС* – ставка налога на добавленную стоимость (*ННДС* = 18%).



Расчет цены предложения (*Цпр*) разработанного программного продукта может быть произведен следующим образом:

 (18)



На рынке программных продуктов стоимость разработанного программного продукта будет определяться не ценой предложения (то есть – минимальной ценой, за которую производитель готов продать свой товар), а ценой спроса (то есть – максимальной ценой, за которую потребитель готов купить данный товар). Следовательно, для того, чтобы скомпенсировать все затраты на разработку программного продукта и получить установочную прибыль, необходимо тиражировать разработанный программный продукт и реализовать не менее определенного расчетного количества копий. Это и является основным условием окупаемости.

Тогда, расчет минимального количества копий (*К*min), необходимых для реализации, можно выполнить следующим образом:

 (19)

где *Цспр* – цена спроса на подобные программные продукты (25000руб./ед.);

*ЗТК* – затраты на тиражирование одной копии разработанного программного продукта (руб./ед.).

В свою очередь затраты на тиражирование одной копии разработанного программного продукта рассчитываются как:

 (20)

где *СН* – стоимость носителя, на котором записывается программный продукт (15руб./шт.);

*ТТК* – время, необходимое для производства одной копии (0,4ч)





Вывод: Таким образом, проведенные расчеты показали: для того, чтобы возместить все произведенные затраты в размере 209438,775 рублей и получить установочную прибыль в размере 41887,755 рублей, необходимо реализовать не менее 12 копий разработанного программного продукта.