**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ “ЛЭТИ”)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 27.03.04 – Управление в технических  системах | |
| **Профиль подготовки** | Системы и технические средства автоматизации и управления | |
| **Факультет** | **ФЭА** | |
| **Кафедра** | **САУ** | |
| ***К защите допустить*** |  | |
| Зав. кафедрой |  | Шелудько В.Н. |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**БАКАЛАВРА**

**Тема: АНАЛИЗ ТЕКСТОВОЙ ИНФОРМАЦИИ С ИСПОЛЬЗОВАНИЕМ НЕЛИНЕЙНЫХ СЕТЕЙ**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Баликов С.В. |
|  |  | *подпись* |  |  |
| Руководитель |  |  |  | Соколов П.В. |
|  |  | *подпись* |  |  |
| Консультант  по Доп. разделу | к.т.н. |  |  | Иванов А.Н. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |
| Консультант  по нормоконтролю | к.т.н., доцент |  |  | Лямкин А.А. |
|  | *(Уч. степень, уч. звание)* | *подпись* |  |  |

Санкт-Петербург

2016

**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой САУ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Шелудько В.Н. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Студент(ка) | Баликов С.В. | | | |  | Группа | 2491 |
| Тема работы: Анализ текстовой информации с использованием  нелинейных сетей | | | | | | | |
| Место выполнения ВКР: кафедра САУ | | | | | | | |
| Исходные данные (технические требования):  Разработать алгоритм и программу тематической классификации коллекции текстов с использованием нейронных сетей | | | | | | | |
| Содержание ВКР:  Введение, Теоретический обзор методов классификации текстов, Разработка алгоритма векторного квантования, Разработка программного обеспечения и анализ результатов, Безопасность жизнедеятельности, Заключение, Список литературы | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал. | | | | | | | |
| Дополнительные разделы: Безопасность жизнедеятельности | | | | | | | |
|  | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | |
| «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | «\_\_\_» июня 20\_\_\_ г. | | | | |
|  | | |  | | | | |
| Студент(ка) | |  | | Баликов С.В. | | | |
| Руководитель к.т.н. | |  | | Соколов П.В. | | | |
| Консультант  от предприятия | |  | |  | | | |

*(Уч. степень, уч. звание)*

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | **Утверждаю** |
|  | Зав. кафедрой САУ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Шелудько В.Н. |
|  | «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент(ка) | | Баликов С.В. |  | Группа | | 2491 |
| Тема работы: Анализ текстовой информации с использованием нелинейных сетей | | | | | | |
| № п/п | Наименование работ | | | | Срок выполнения | | |
| 1 | Анализ существующих подходов и разработка алгоритма | | | | 14.02 – 29.02 | | |
| 2 | Разработка программного обеспечения | | | | 01.03 – 31.03 | | |
| 3 | Отладка ПО | | | | 01.04 – 30.04 | | |
| 4 | Проведение исследований эффективности алгоритма  с различными настройками | | | | 01.05 – 31.05 | | |
| 5 | Наименование Дополнительного раздела | | | | 01.05 – 23.05 | | |
| 6 | Оформление пояснительной записки | | | | 10.05 – 01.06 | | |
| 7 | Оформление иллюстративного материала | | | | 01.06 – 08.06 | | |

|  |  |  |
| --- | --- | --- |
| Студент(ка) |  | Баликов С.В. |
| Руководитель | к.т.н. | Соколов П.В. |
| Консультант  от предприятия |  |  |

*(Уч. степень, уч. звание) Подпись*

РЕФЕРАТ

Данная выпускная квалификационная работа посвящена разработке структуры нейронной сети и алгоритма её функционирования для задачи классификации текстов.

В первой главе проведён теоретический обзор методов классификации текстов, дано понятие векторной модели текста и кластерного анализа. В данной главе были рассмотрены типы нейронных сетей, а также структура отдельного класса – нейронных сетей Кохонена.

Во второй главе были рассмотрены алгоритмы предварительной фильтрации и векторного квантования, а также их математическая модель.

В третьей главе было описано разработанное программное обеспечение с учётом структуры и используемых методов, а также проведены исследования и анализ результатов.

В результате выполнения квалификационной работы были созданы два программных модуля: первый модуль выполняет задачу предварительной фильтрации и подготовке структур входных данных; второй модуль выполняет задачу обучения нейронной сети с дальнейшим выводом результатов вычислений.

В результате отладки программы были достигнуты оптимальные значения варьируемых параметров, обеспечивающих безошибочную работу программы. Таким образом, процент правильного распознавания текстов составил 90,38%, что является очень хорошим результатом.

Пояснительная записка включает в себя 66 страниц текста, 13 рисунков.

ABSTRACT

The graduate work is devoted to development of neural net's structure and an algorithm of its functioning. The neural net is used for classification of texts' collection by their topics. The total amount of chapters in the graduate work is three.

The first chapter contains the total theoretical overview of clusterization methods. The definition of cluster analysis and vector text model is given. In addition first chapter describes types of neural nets and structure of its particular class - Kohonen neural net.

The second chapter contains overview of prefiltration algorithms and vector quantization algorithms with their own mathematical model.

The third chapter describes developed software with its structure and methods. Also there is a total analysis of results.

As a result two program modules were developed: first module performs the task of prefiltration and preparation structures of input data; second module has a goal of training the neurons in the neural net and calculating texts' clusters as result.

The graduate work includes 66 pages of text, 13 figures.

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ 8

ВВЕДЕНИЕ 9

1. ТЕОРЕТИЧЕСКИЙ ОБЗОР МЕТОДОВ КЛАССИФИКАЦИИ ТЕКСТОВ 10

1.1 Векторная модель текста 10

1.2 Кластерный анализ 11

1.3 Требования к разрабатываемому алгоритму. Достоинства и недостатки методов кластеризации 12

1.4 Типы нейронных сетей 14

1.5 Нейронная сеть Кохонена 16

2. РАЗРАБОТКА АЛГОРИТМА ВЕКТОРНОГО КВАНТОВАНИЯ 18

2.1 Алгоритм предварительной фильтрации 18

2.2 Стандартный алгоритм векторного квантования 19

2.2.1 Векторная модель текста и входные данные сети 19

2.2.2 Алгоритм обучения сети 21

2.3 Упрощенный метод классификации текстов 25

2.3.1 Векторная модель текста и входные данные сети 25

2.3.2 Алгоритм обучения сети 29

3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И АНАЛИЗ РЕЗУЛЬТАТОВ 32

3.1 Общие требования к разработке программного обеспечения 32

3.2 Программа подготовки исходных данных Qtdes 34

3.2.1 Структура программы Qtdes 34

3.2.2 Функционирование программы Qtdes 38

3.3 Программа, реализующая нейронную сеть, NeuralNetClassificator 42

3.3.1 Структура программы NeuralNetClassificator 42

3.3.2 Функционирование программы NeuralNetClassificator 45

3.4 Заключительные исследования и анализ результатов 46

3.4.1 Анализ выполнения предварительной фильтрации 46

3.4.2 Анализ обучения 48

4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ 52

ЗАКЛЮЧЕНИЕ 55

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 56

ПРИЛОЖЕНИЕ А 57

ПРИЛОЖЕНИЕ Б 60

# ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

LSI (англ. Latent Semantic Indexing) - скрытое семантическое индексирование;

TF (англ. term frequency) - частота слова;

IDF (англ. inverse document frequency) - обратная частота документа;

Виджет (англ. widget) - элемент интерфейса;

Парсинг (англ. parcing) - процесс сопоставления линейной последовательности лексем естесственного или формального языка с его формальной грамматикой

# ВВЕДЕНИЕ

Применение и совершенствование средств вычислительной техники открывает перед человеком спектр новых возможностей, однако вместе с этим появляется ряд других проблем, связанных с ростом обрабатываемых массивов полнотекстовых документов. При работе с полнотекстовыми базами данных возникает задача поиска и классификации документов по их содержанию. На сегодняшний день можно привести в пример множество методов поиска, в том числе реализуемых поисковыми машинами в Интернете, однако в некоторых случаях эти методы могут некорректно выбирать информацию по запросу пользователя.

В связи с этим в данной работе ставится задача разработки поискового алгоритма, который выполняет поиск в заранее подготовленной базе данных с использованием нелинейных сетей, а именно нейронных сетей Кохонена.

В качестве основного инструмента использовалась среда разработки QT Creator с использованием технологии C++.

Целью ВКР является разработка структуры нейронной сети и алгоритма её функционирования. Для достижения цели необходимо решить ряд задач:

* провести теоретический обзор методов классификации текстов;
* разработать алгоритмы предварительной фильтрации и векторного квантования;
* разработать программное обеспечение с учётом структуры и используемых методов;
* провести исследования и выполнить анализ результатов.

# 1. ТЕОРЕТИЧЕСКИЙ ОБЗОР МЕТОДОВ КЛАССИФИКАЦИИ ТЕКСТОВ

## 1.1 Векторная модель текста

В конце 80-х годов была предложена векторная модель как альтернатива лексическому бесконтекстному индексированию. В простейшем случае векторная модель предполагает сопоставление каждому документу частотного спектра слов и соответственно вектора в лексическом пространстве. В процессе поиска частотный портрет запроса рассматривается как вектор в том же пространстве и по степени близости (расстоянию или углу между векторами) определяются наиболее релевантные документы. В более продвинутых векторных моделях размерность пространства сокращается отбрасыванием наиболее распространенных или редко встречающихся слов, увеличивая тем самым процент значимости основных слов. Главным достоинством векторной модели является возможность поиска и ранжирования документов по подобию, то есть по их близости в векторном пространстве. Однако практика показывает, что при оценке близости запроса к документу результаты поиска могут быть не всегда удовлетворительными, что особенно проявляется, когда запрос содержит малое количество слов. Для получения лучшей релевантности отклика в 1990 году была предложена модель скрытого семантического индексирования ― Latent Semantic Indexing (LSI) [1]. Модель использовала Singular Value Decomposition (SVD) для перехода от разреженной матрицы слов к компактной матрице главных собственных значений. LSI показала значительное превосходство в результатах поиска по сравнению с лексическим методом, однако сложность модели часто приводила к существенному проигрышу в скорости на больших коллекциях документов по сравнению с традиционной булевой техникой. Одна из наиболее работоспособных систем на основе LSI была создана в Беркли в 1995 году Майклом Берри и Тодом Летче [1].

## 1.2 Кластерный анализ

Кластерный анализ (англ. cluster analysis) — многомерная статистическая процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы. Задача кластеризации относится к статистической обработке, а также к широкому классу задач обучения без учителя.

Кластерный анализ выполняет следующие основные задачи:

1. Разработка типологии или классификации.
2. Исследование полезных концептуальных схем группирования объектов.
3. Порождение гипотез на основе исследования данных.
4. Проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

Независимо от предмета изучения применение кластерного анализа предполагает следующие этапы:

1. Отбор выборки для кластеризации. Подразумевается, что имеет смысл кластеризовать только количественные данные.
2. Определение множества переменных, по которым будут оцениваться объекты в выборке, то есть признакового пространства.
3. Вычисление значений той или иной меры сходства (или различия) между объектами.
4. Применение метода кластерного анализа для создания групп сходных объектов.
5. Проверка достоверности результатов кластерного решения.

В качестве целей кластеризации выделяют:

1. Понимание данных путём выявления кластерной структуры. Разбиение выборки на группы схожих объектов позволяет упростить дальнейшую обработку данных и принятия решений, применяя к каждому кластеру свой метод анализа (стратегия «разделяй и властвуй»).
2. Сжатие данных. Если исходная выборка избыточно большая, то можно сократить её, оставив по одному наиболее типичному представителю от каждого кластера. В данной работе используется как основной механизм ускорения процесса поиска, что позволяет избежать большой вычислительной сложности процесса обработки поискового запроса.

Во всех этих случаях может применяться иерархическая кластеризация, когда крупные кластеры дробятся на более мелкие, те в свою очередь дробятся ещё мельче, и т. д. Такие задачи называются задачами таксономии. Результатом таксономии является древообразная иерархическая структура. При этом каждый объект характеризуется перечислением всех кластеров, которым он принадлежит, обычно от крупного к мелкому [1].

## 1.3 Требования к разрабатываемому алгоритму. Достоинства и недостатки методов кластеризации

Разрабатываемая система должна удовлетворять нижеперечисленным требованиям:

1. Улучшение быстродействия системы
2. Точность и эффективность вычислений
3. Рациональность использования ресурсов системы
4. Обеспечение точности поиска и соответствие результатов поиска пользовательскому запросу
5. Функциональная полнота системы
6. Автономность и устойчивость разработанной системы, а также безошибочность ее работы

Необходимо поговорить о методах кластеризации, однако общепринятой классификации данных методов не существует. Поэтому выделяют определенные группы подходов, характерных для методов.

Особого внимания заслуживает вероятностный подход, в который входит метод К-средних. В данном методе все объекты в равной степени относятся соответственно к k классам.

В основе K-means (К-средних) лежит итеративный процесс стабилизирования центроидов кластеров. Основной характеристикой кластера является его ценроид и вся работа алгоритма направлена на стабилизирование или, в лучшем случае, полное прекращение изменения центроида кластера.

Достоинства метода:

1. Низкая вычислительна сложность – O(knT), где n – число документов, k – число кластеров, T – количество итераций.
2. Метод не нуждается в обучении и при необходимости может накапливать сведения для дальнейшего увеличения точности работы.
3. Использование Байесовских оценок параметров кластеризации.

Недостатки метода:

1. Требуется задание количества кластеров, как минимум на начальных этапах – до использования априорной информации.
2. Сравнительно низкая точность.
3. В том случае, когда центроиды кластеров выбираются случайным образом, результаты, получаемые над одной и той же выборкой документов, будут отличаться. Это может происходить по причине неудовлетворительной работы генератора случайных чисел и вследствие равномерного распределения документов в пространстве.
4. Без явных областей сгущения[2];

Таким образом, алгоритм весьма чувствителен к выбору начальных центров кластеров. Так как кластеры выбираются случайно, то из-за этого возникает погрешность. С этой проблемой успешно справляется алгоритм нечеткой кластеризации C-means. В основе данного метода лежат системы искусственного интеллекта. В данном методе присутствующее множество векторов требуется разбить на заданное число нечетких множеств. Также используется матрица принадлежности, которая определяет принадлежность элемента исходного множества векторов какому-то кластеру. Данный метод определяет вероятность того, что объект принадлежит к тому или иному кластера, а не просто однозначно дает однозначный ответ.

## 1.4 Типы нейронных сетей

С инженерной точки зрения нейронная сеть представляет собой сильно распараллеленную динамическую систему с топологией направленного графа, которая может выполнять переработку информации посредством изменения своего состояния в ответ на постоянный или импульсный входной сигнал.

Конкретный вид выполняемого сетью преобразования информации обусловливается не только характеристиками нейроподобных элементов, но и особенностями ее архитектуры, т. е. той или иной топологией межнейронных связей, выбором определенных подмножеств нейроподобных элементов для ввода и вывода информации, наличием или отсутствием конкуренции, направлением и способами управления и синхронизации информационных потоков между нейронами и т. д. При этом следует отметить некоторые свойства НС: возможности сети возрастают с увеличением числа слоев сети и количества связей между ними; введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети; сложность алгоритмов функционирования сети (в том числе, например, введение нескольких типов синапсов – возбуждающих, тормозящих и др.) также способствует усилению мощи НС.

Так как проблема синтеза НС сильно зависит от решаемой задачи, дать общие подробные рекомендации затруднительно. В большинстве случаев оптимальный вариант получается на основе интуитивного подбора.

Выделяют несколько (обычно три) основных типов нейронных сетей, отличающихся структурой и назначением:

1. Иерархические сети. Информация в таких сетях передается в процессе последовательного перехода от одного уровня иерархии к другому. Нейроны образуют два характерных типа соединений — конвергентные, когда большое число нейронов одного уровня контактирует с меньшим числом нейронов следующего уровня, и дивергентные, в которых контакты устанавливаются с большим числом нейронов последующих слоев иерархии.
2. Локальные сети, формируемые нейронами с ограниченными сферами влияния. Нейроны локальных сетей производят переработку информации в пределах одного уровня иерархии. При этом функционально локальная сеть представляет собой относительно изолированную тормозящую или возбуждающую структуру.
3. Важную роль также играют так называемые дивергентные сети с одним входом. Командный нейрон, находящийся в основании такой сети может оказывать влияние сразу на множество нейронов, и поэтому сети с одним входом выступают согласующим элементом в сложном сочетании нейросетевых систем всех типов.

В зависимости от используемой в НС выходной функции нейрона различают бинарные и аналоговые сети. Первые из них оперируют с двоичными сигналами, и выход каждого нейрона может принимать только два значения: логический ноль ("заторможенное" состояние) и логическая единица ("возбужденное" состояние). В аналоговых сетях выходные значения нейронов способны принимать непрерывные значения при замене ступенчатой (пороговой) функции сигмоидной [3].

## 1.5 Нейронная сеть Кохонена

Для обеспечения работы нейронной сети Кохонена требуется выполнение целого ряда условий. Для начала объект классификации требуется представить в виде некого входного вектора, который подаётся на вход нейронной сети. Общее количество элементов входного вектора определяет количество нейронов входного слоя, а количество классов - нейронов выходного слоя, то есть каждый нейрон из выходного слоя отвечает за свой класс. Таким образом, значения, принимаемые нейронами выходного слоя, отображают насколько входной вектор объекта принадлежит соответствующему классу. Величина значения, которое принимает нейрон, характеризует степень принадлежности.

Однако существует другой метод реализации нейронной сети - "победитель забирает все". В данном случае значения выходных нейронов могут принимать значения либо нуля, либо единицы. Также следует отметить, что для одного вектора может быть только 1 нейрон, значение которого равно 1, а значения остальных нейронов, очевидно, равны 0.

Нейронные сети Кохонена – класс нейронных сетей, основным элементом которых является слой Кохонена.

Слой Кохонена состоит из параллельно действующих адаптивных линейных элементов. Все они имеют одинаковое число входов и получают на свои входы один и тот же вектор входных сигналов .

На выходе -го линейного элемента получаем сигнал:

, (1.1)

где — весовой коэффициент -го входа -го нейрона,  — пороговый коэффициент.

После прохождения слоя линейных элементов сигналы посылаются на обработку по правилу «победитель забирает всё»: среди выходных сигналов ищется максимальный. При этом его номер равен . Окончательно, на выходе сигнал с номером равен единице, остальные — нулю. Если максимум одновременно достигается для нескольких , то либо принимают все соответствующие сигналы равными единице, либо только первый в списке (по соглашению)[1].

Вывод: в данной главе было дано понятие векторной модели текста и кластерного анализа. Были рассмотрены основные типы нейронных сетей, а также методы кластеризации с учётом их достоинств и недостатков, целесообразности их использования. Основное внимание было уделено отдельному классу нейронных сетей, а именно нейронным сетям Кохонена с их собственной структурой и функциональностью.

# 2. РАЗРАБОТКА АЛГОРИТМА ВЕКТОРНОГО КВАНТОВАНИЯ

## 2.1 Алгоритм предварительной фильтрации

Задача алгоритма предварительной фильтрации заключается в отсеве широкоупотребительных слов из текстовой коллекции документов. Для осуществления фильтрации необходимо выполнение следующих условий:

1. Наличие составленной коллекции документов в структуре QMap <QString, double> DocumentCollection, содержащая в себе список всех слов коллекции, где каждому слову соответствует частота. Данная частота - это частота встречаемости слова в коллекции;
2. Наличие порога фильтрации, который может фиксироваться исключительно пользователем;

Используя программные средства, необходимо осуществить подсчёт частоты встречаемости слов. Реализуется это следующим способом. Подсчитывается количество документов в коллекции NDC. Пусть частота встречаемости слова будет иметь размерность в процентах. При подокументном сканировании все слова, которые хотя бы один раз встречаются в документе, должны увеличить своё значение в структуре DocumentCollection слов на одну и туже же величину. Причём сложение происходит один раз при сканировании одного документа и при переходе к следующему документу также может складываться только единожды. Следовательно, количество операций сложения не должно превышать величину NDC, то есть количество документов коллекции.

Таким образом, данное приращение представляет собой частоту встречаемости слова в коллекции, при условии если оно встретилось 1 раз из NDC документов. Математическое выражение выглядит следующим образом:

(2.1)

где NDC - количество документов в коллекции.

После прогона всех NDC документов получаем готовую структуру всех слов в коллекции с имеющимися частотами встречаемости, которые требуется сравнить с заданным пользователем порогом фильтрации. В процессе операции сравнения из всей коллекции удаляются слова, соответствующие условию:

(2.2)

где - задаваемый пользователем порог фильтрации.

Все слова, удовлетворяющие условию 2.2, принято считать широкоупотребительными для текстовой коллекции.

## 2.2 Стандартный алгоритм векторного квантования

### 2.2.1 Векторная модель текста и входные данные сети

В данной работе будут рассмотрены два метода классификации текстов, а именно так называемые стандартный и упрощенный их виды. Также следует немного поговорить о целесообразности использования данных методов с учетом их плюсов и минусов. В основе данных методов лежит принцип построения векторной модели текста, то есть сопоставление каждому документу вектора, находящегося в лексическом пространстве.

При использовании стандартного метода классификации текстов требуется определить коэффициенты значимости слова в каждом документе типа TF-IDF.

Статистическая мера TF-IDF помогает оценить важность определенного слова в контексте документов текстовой коллекции. На вес слова большое влияние оказывает число употребления слова в документе. При употреблении данного слова в других документах коллекции его вес уменьшается. Мера TF-IDF часто используется для анализа текстов больших текстовых коллекций. Ниже мы рассмотрим структуру формулы данной статистической характеристики:

1. TF (term frequency - частота слова) - это отношение числа вхождения k-ого слова в j-ый документ общему количеству слов в данном документе. Данная характеристика помогает оценить важность k-ого слова в пределах j-ого документа.

, (2.3)

где - число вхождений слова в документ ; - общее число слов в документе .

1. IDF (inverse document frequency - обратная частота документа) - инверсия частоты, с которой некоторое слово встречается в документах коллекции. Данная характеристика помогает уменьшить вес слов, часто встречающихся в коллекции документов.

(2.4)

где - количество документов коллекции; - количество документов, в которых встречается слово .

Требуется заметить, что в данной формуле k - сквозное для всех документов в коллекции. Это значит, что k находится в диапазоне . Количество слов позднее может оказаться меньше после предварительной фильтрации. Таким образом, для каждого уникального слова в пределах всей коллекции существует только одно.

Таким образом, чтобы получить меру требуется перемножить коэффициенты, то есть:

(2.5)

Эта мера позволяет получить большой вес широко-употребляемому слову в пределах одного документа, и низкий во всех остальных.

В итоге в качестве входных данных для обучения сети будем иметь коллекцию документов , в которых заранее определена тема (в целях последующей проверки); номер кластера (нейрона), который будет определен в результате работы сети; наборы для оставшихся после фильтрации слов .

Ранее было решено использовать сеть векторного квантования как разновидность сети Кохонена, реализующей алгоритм K-средних (K-means) кластерного анализа. Сеть однослойная. Число нейронов принимается равным числу тем.

При инициализации входных данных требуется ввести первоначальные центры кластеров. Для этого требуется посчитать средние . Затем найти 6 документов, для которых расстояние максимально.

### 2.2.2 Алгоритм обучения сети

В ходе обучения требуется по одному документу последовательно анализировать коллекцию слов с заданными весами. В итоге для одного документа осуществляется проверка всех тем. На Рисунке 2.1 изображена схема сети стандартного метода классификации текстов.

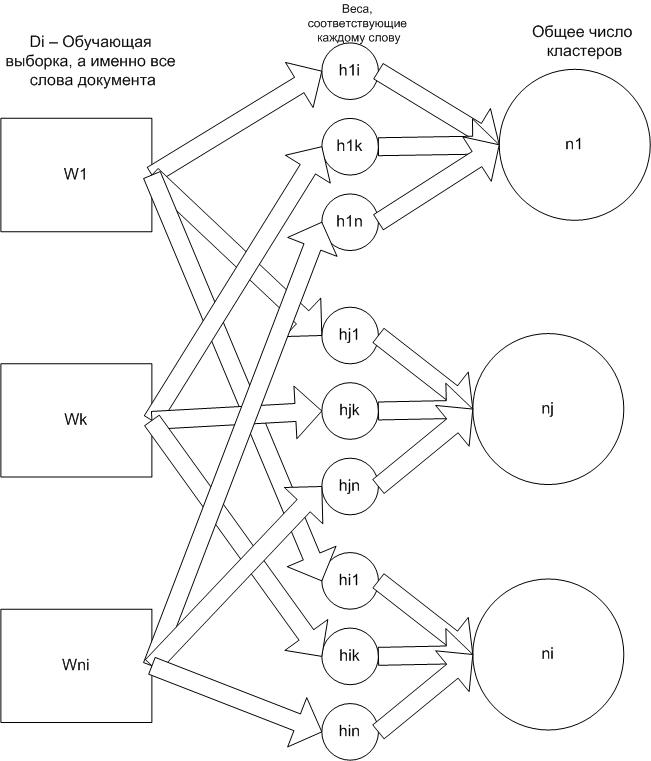


Рисунок 2.1 - Схема сети стандартного метода для случая одного документа

Сначала в этом документе рассчитывается расстояние . Определяется оно по следующей формуле:

(2.6)

Затем определяем победителя для одного документа. Требуется выбрать минимальное .

(2.7)

После произведенных операций производится настройка нейрона-победителя.

Делается это путем элементарных алгебраических преобразований и введением трех коэффициентов , , .

, (2.8)

- постоянный коэффициент, который определяется опытным путем,

, (2.9)

где -максимальное количество шагов в обучении; - текущий шаг в обучении, всегда меньше 1.

Затем происходит увеличение весов слов, встречающихся в документе, нейрона-победителя по следующей формуле:

Если , то фиксируем .

В случае обучения остальных (проигравших) нейронов производим уменьшение весов слов нейрона, которые содержатся в анализируемом документе.

Данное преобразование производится введение следующих коэффициентов:

, (2.11)

- постоянный коэффициент, который определяется опытным путем,

(2.12)

Настройка весов нейронов производится при помощи следующих математических преобразований:

(2.13)

Следующим шагом выбирается следующий обучающий вектор и нам ним опять производятся вышеперечисленные операции.

В ходе процесса обучения нейронов требуется ввести несколько дополнительных модификаций:

1. Обеспечить вывод , чтобы иметь возможность построить график сходимости.
2. Оценивать - коэффициент правильного распознавания, где -правильно определенные темы документа, - неправильно определенные темы документа.
3. Если слова нет в документе, но есть в кластере, то есть , но , то в дополнение к формулам тренировки нейронов производим уменьшение весов слов нейрона по формуле:

(2.14)

Уменьшение весов требуется производить с учетом .

## 2.3 Упрощенный метод классификации текстов

### 2.3.1 Векторная модель текста и входные данные сети

Прежде всего, требуется ввести несколько важных обозначений, которые будут встречаться в дальнейшем:

1. С - коллекция документов; NDC - число документов коллекции
2. - это номер темы. Максимальное количество тем - 6.
3. - номер документа для j-ой темы.
4. - k-ое слово i-го документа.
5. -общее число слов коллекции(данное число равно числу входов нейрона сети)
6. t - шаг обучения. Это число обозначает число циклов подачи всей коллекции документов на вход нейронной сети.
7. -коэффициент (вес) j-ого нейрона для k-ого входа, где . Вес одного слова никогда не должен превышать 1и быть ниже 0.
8. Введем переменную , представляющую собой сумму весовых коэффициентов для нейрона - победителя и подчиняющуюся условиям

(2.15)

(2.16)

Введем переменную , представляющую собой отношение расстояния для нейрона - победителя к сумме расстояний всех 6 нейронов.

(2.17)

Существует несколько кардинальных отличий модифицированного "упрощенного" алгоритма от "полного".

В данном методе вместо коэффициентов значимости слова типа IFIDF используются бинарные коэффициенты, при которых:

Таким образом, получим вектор выходов нейронной сети:

, (2.18)

где X - бинарный вектор входных сигналов; H - матрица весовых коэффициентов.

Следовательно, на выходе каждого нейрона в случае подачи вектора слов одного документа:

1. Для всех слов документа подсчитывается расстояние, которое (вместо евклидового расстояния) представляет собой меру близости i-го документа к j-му кластеру. То есть происходит суммирование тех весов слов из общей коллекции, которые встречаются в документе .

(2.19)

1. Веса нейронов пронормированны;

Нормировка осуществляется следующими способами:

* Для определения нейрона - победителя требуется выполнение условия .
* Вводится переменная , описанная выше. Очевидно, что не может превышать 1 и быть меньше 0. Эта величина помогает оценить, насколько i-ый документ ближе к j-му нейрону(кластеру), чем к остальным. Если все нейроны получили одинаковую близость к тексту, то средняя Δ определяется.

Таким образом, можно сформулировать задачу обучения (самообучения). Необходимо увеличить веса для слов, входящих в документы данного кластера j-го нейрона и уменьшить веса слов проигравших нейронов.

На Рисунке 2.2 изображена схема сети упрощенного метода классификации текстов.

В работе решено использовать "упрощенный" метод классификации текстов ввиду существенных преимуществ.

Во-первых, коэффициент значимости слова TFIDF пригоден для больших текстов и больших наборов документов, где достаточны статистические выборки для расчета относительных частот.

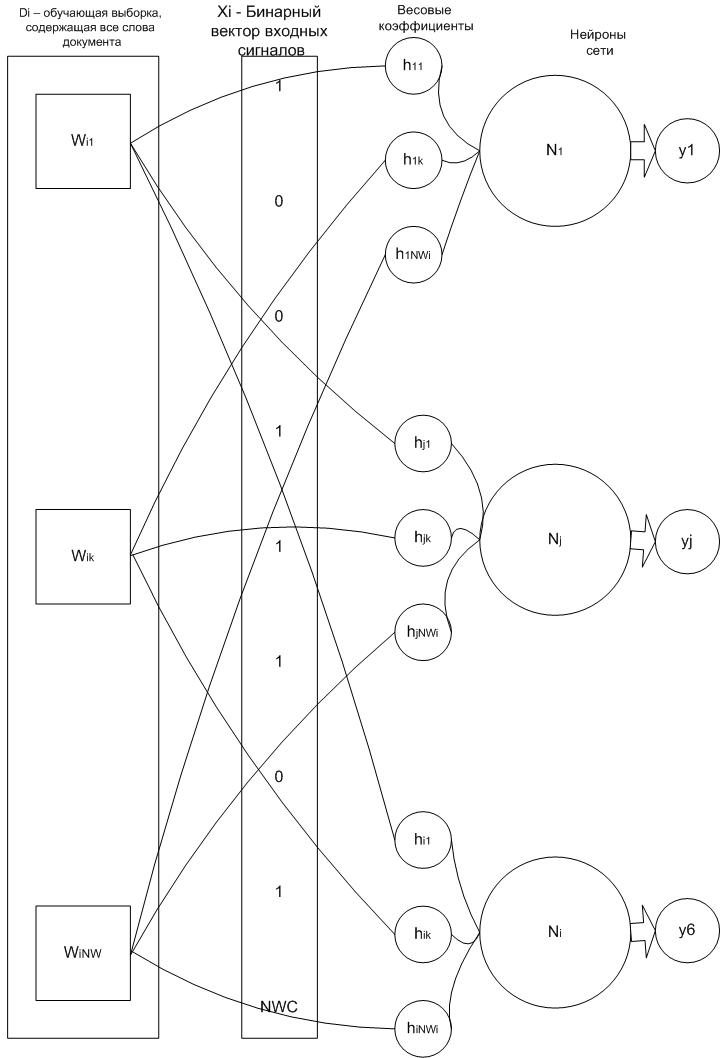


Рисунок 2.2 - Схема сети упрощенного метода для случая одного документа

В данной работе на вход нейронной сети подаётся небольшая коллекция документов (150 шт.) с короткими текстами (). Небольшой размер коллекции делает нецелесообразным определение частот употребления слов. Во-вторых, вычислительные алгоритмы становятся значительно проще. В-третьих, обучающие алгоритмы значительно упрощаются. В-четвёртых, веса нормируются, что позволяет нормировать другие коэффициенты.

### 2.3.2 Алгоритм обучения сети

Обучение сети с использованием упрощенного метода классификации текстов включает в себя как обучение нейрона - победителя (для случая одного документа), так и проигравших нейронов.

В ходе работы алгоритма требуется определять целесообразность выполнения настройки. Данное требование можно реализовать введением величины , с которой сравнивается Δ каждого нейрона. Если , то текст для этого нейрона принято считать неинформативным и настройку производить не требуется. Расстояние незначительно превосходит другие , что говорит о малой выраженности победы. Необходимо следить за выполнением условия:

(2.19)

Величину можно свободно менять по мере трениировки нейронов. Вначале большое, и поэтому побеждать будут нейроны в текстах сильно выраженной тематики. В процессе обучения требуется постепенно снижать , чтобы иметь возможность распределять текстовые документы слабо выраженной тематики с содержащимися в них нехарактерными словами в тот или иной кластер.

Очевидно, что слово, принадлежащее нейрону, но не принадлежащее документу, свой вес менять не будет. Если же слово содержится в , то требуется увеличить вес нейрона-победителя по следующей формуле:

, (2.20)

где - коэффициент, обеспечивающий скорость настройки;

Коэффициент требуется увеличивать в процессе обучения. При . Величина в данной формуле позволяет увеличивать шаг настройки в текстах сильно выраженной тематики. Соответственно, чем меньше, тем меньше шаг настройки.

Следствия нормировки :

1. Коэффициент увеличивается по ходу настройки

(2.21)

Поскольку , то есть . Следовательно, коэффициент растет в процессе обучения.

1. Так как , то , следовательно, обеспечивается . Таким образом, коэффициент не превышает 1.

Модификацией алгоритма является процесс обучения не только победившего нейрона, но и остальных проигравших нейронов. Очевидно, что для нейрона-победителя мы увеличивали коэффициенты, то, аналогично, для проигравших нейронов коэффициенты требуется уменьшать. Так же как и в алгоритме тренировки победителя требуется вводить порог настройки.

В случае отсутствия слова в документе , то его вес не меняется. Если слово принадлежит документу и его вес у проигравшего нейрона , то необходимо провести уменьшение весов по следующей формуле

*,* (2.22)

где - коэффициент, обеспечивающий скорость настройки.

Коэффициент требуется увеличивать в процессе . Величина находится в пределах .

Следствия нормировки :

1. Коэффициент убывает во время настройки

(2.23)

Поскольку , то , следовательно,

1. может быть < 0, поэтому при выполнении этого условия необходимо при фиксировать .

Вывод: в данной главе были рассмотрены несколько алгоритмов с соответствующими им математическими моделями. Алгоритм предварительной фильтрации используется в дальнейшем для подготовки структур входных данных текстовой коллекции. Также был проведён анализ двух алгоритмов векторного квантования: стандартного и упрощённого. Ввиду наличия относительно небольшой текстовой коллекции было решено использовать упрощенный метод векторного квантования, что незначительно упрощает вычислительные и обучающие алгоритмы программы.

# 3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И АНАЛИЗ РЕЗУЛЬТАТОВ

## 3.1 Общие требования к разработке программного обеспечения

Существует понятие специфики разработки программных средств, которое включает в себя ряд характерных особенностей. Во-первых, требования к программному обеспечению носят неформальных характер, однако сам объект разработки, программное обеспечение, формализован. Таким образом, можно наблюдать своеобразный переход от неформального к формальному.

Во-вторых, следует отметить, что разработка программного обеспечения - дело творческого характера. Это значит, что выполнение программы не включает в себя выполнение одного заданного алгоритма, последовательности, а требует пользовательского выбора, решения.

Требования к программному обеспечению - совокупность утверждений относительно атрибутов, свойств или качеств программной системы, подлежащей реализации. Создаются в процессе разработки требований к программному обеспечению, в результате анализа требований [1].

Существует три вида требований к разработке программного обеспечения:

1. Бизнес - требования - это требования, которые включают в себя назначение программного обеспечения. Данные требования описываются в регламентированных документах, а именно документов о границах проекта и видения;
2. Пользовательские требования - это требования, представляющие список задач пользователя, которые реализует программа, а также способы решения данных задач;
3. Функциональные требования - это требования, предопределяющее поведение системы и функции, которые данная система может выполнять. Функциональные требования описываются в системной спецификации;

Все вышеперечисленные требования носят исключительно функциональный характер, которые описывают требования к поведению системы.

Перечислим основные функциональные требования разрабатываемого программного обеспечения:

1. Улучшение быстродействия системы
2. Точность и эффективность вычислений
3. Рациональность использования ресурсов системы
4. Обеспечение точности поиска и соответствие результатов поиска пользовательскому запросу
5. Функциональная полнота системы
6. Автономность и устойчивость разработанной системы, а также безошибочность ее работы

В данной главе рассмотрены два программных модуля, выполняющих различные функции. Первая программа осуществляет приведение входных исходных данных в надлежащий вид, а именно парсинг текстов на слова и занесение их в соответствующие структуры. Помимо всего этого программа осуществляет подсчёт слов в каждом документе и всей текстовой коллекции в целом. Также в процессе парсинга осуществляется подсчёт частоты встречаемости слова в документах коллекции. С полученной частотой происходит сравнение задаваемого пользователем порога фильтрации. Если частота слова превышает величину порога, то данное слово удаляется из текстовой коллекции и считается широкоупотребительным.

Второй программный модуль обеспечивает работу алгоритма обучения нейронной сети. В нём представлены все самые основные математические операции, например расчёт характерных расстояний, увеличение и уменьшение весов соответственно победивших и проигравших нейронов (обучение нейронов) и другие операции, соответствующие методике, описанной в пункте 2.3. Также производится анализ полученных результатов.

## 3.2 Программа подготовки исходных данных Qtdes

### 3.2.1 Структура программы Qtdes

Программа подготовки исходных данных представляет собой программный модуль, который осуществляет парсинг текстов коллекции и предварительную фильтрацию слов. Функциональность данного модуля включает в себя:

1. Интерфейсная часть, которая отвечает за считывание текстов из задаваемых пользователем директорий;
2. Разбиение текстовых потоков (парсинг слов) на элементы (слова, знаки и т.д.;
3. Предварительная фильтрация входных данных, а именно текстовой информации. В результате происходит удаление символов переноса строки, табуляции, абзацев, грамматических символов, цифровых данных, нечитаемых символов. Таким образом остаются одни слова, причем важно программно реализовать приведение всех слов текстовой коллекции к нижнему регистру;
4. Фильтрация общеупотребительных слов. После проведения фильтрации в документах остаются только термины, то есть слова специфические для заданных тем. В интерфейсной части можно регулировать коэффициент фильтрации и, следовательно, пропускную способность алгоритма фильтрации;
5. Подсчёт статистической информации, который включает в себя подсчёт количества слов, частоты слов в документах и т.д.;
6. Размещение полученных данных, а именно наборы слов для документов и коллекции, в специальные структуры данных;
7. Вывод информации в файлы путем задания директории в интерфейсной части;

Принятые обозначения возьмем аналогично пункту 2.2.1 .

Структура программы включает в себя несколько важных программных модулей. Первый главный модуль DocumentsCollectionPreprocessing имеет следующую функциональность:

1. Запуск программы;
2. Запуск интерфейсной части;
3. Запуск всех методов для реализации функциональности программы;
4. Хранение данных в соответствующих структурах;

Интерфейсная часть разрабатываемого программного модуля должна обладать следующей функциональностью:

1. Кнопка START для запуска расчётной части программы;
2. Виджет FileDialog для задания директорий с файлами для данной темы;
3. Выпадающее меню для задания темы. Также следует обеспечить защиту, то есть сделать недоступным выбор новой директории без задания названия темы;
4. Текущее состояние программы должно регулироваться соответствующим виджетом ProgressBar или похожим на него;
5. Виджет FileDialog для задания директории с файлами результатов вычислений;
6. Приложение должно иметь стандартные средства работы с окном интерфейсной части, а именно иконки закрытия программы, свертывания/развертывания окна, перетаскивания окна, окно задания .

Функциональные модули включают в себя классы для реализации функциональности программы. Количество и структура данных модулей также зависят от функциональности. Желательно, чтобы каждому пункту функциональных требований соответствовал свой функциональный модуль (класс). Все эти модули обязательно должны запускаться по мере надобности в ходе работы программы из главного модуля DocumentsCollectionPreprocessing.

Необходимо изобразить структуру директорий входных и выходных данных. На Рисунке 3.1 изображена структура директории с входными данными

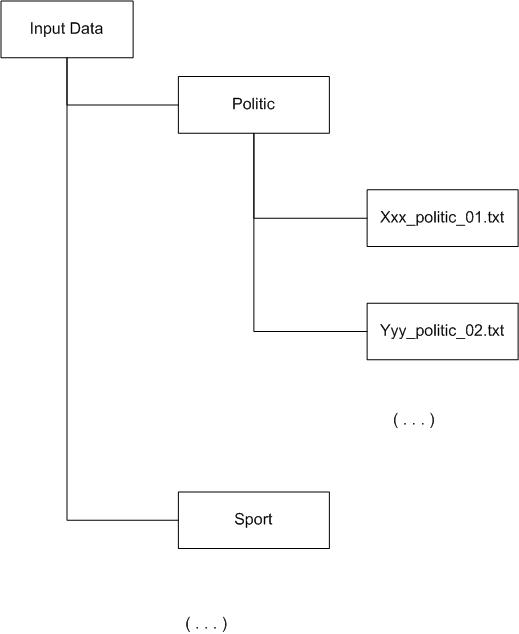
****

Рисунок 3.1 - Структура директории входных данных

На Рисунке 3.2 изображен пример структуры директории с выходными данными.

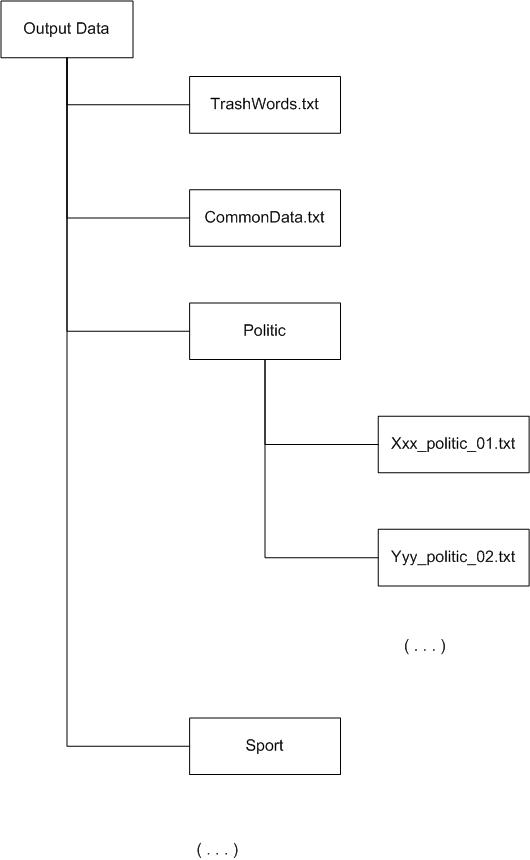
****

Рисунок 3.2 - Структура директории выходных данных

Структуры хранения представляют собой два класса:

1. Класс DocumentsCollection имеет следующие поля

* map (QMap <QString, double>) WordsCollection - это ассоциативный список для хранения слов в коллекции (ключ) и частоты упоминаемости документов в коллекции;
* vector (<Document>) documentsVector - вектор для хранения объектов класса Document;
* int NT = 6 - количество тем и нейронов;
* int NDC = 150 - число документов в коллекции;
* int NWC - количество слов в коллекции (размер WordsCollection);
* map (QMap <QString, double>) WordsTrash - карта всех отфильтрованных из WordsCollection слов;
* Класс Document - класс, содержащий информацию, относящуюся к конкретному тексту (документу) имеет следующие поля:
* documentName (<QString>) - имя документа;
* documentType (<QString>) - тип документа;
* int ND - число слов в документе;
* map (QMap <QString, double>) wordsDocument - ассоциативный список для хранения слов данного документа (ключ) и числа упоминаний слова в документе;

### 3.2.2 Функционирование программы Qtdes

Выполнение программы исходных данных происходит по заранее предопределённому алгоритму. Начало работы программы характеризуется выполнением следующих пунктов:

1. Происходит запуск программы. Создаётся окно интерфейсной части. Создаются структуры для хранения данных;
2. В интерфейсной части пользователь задаёт директорию с хранящимися в ней документами коллекции при помощи виджета FileDialog. Одновременно с этим он задаёт тему документов. Этот процесс повторяется 6 раз до тех пор, пока все документы коллекции полностью не будут загружены;

Работа алгоритма парсинга слов представлена ниже:

1. Вводимые тексты (текстовые потоки) разбиваются на слова (парсинг). Одновременно с этим удаляются все лишние символы (перенос строки, пробелы, цифры, пунктуационные знаки, символы форматирования и т. д.). Должны остаться только слова в формате QString. Одновременно заглавные символы преобразуются в строчные;
2. Считываемые слова в формате QString складываются в качестве ключей в QMap WordsDocument, таким образом, чтобы был уникальный набор слов;
3. Одновременно с пунктом 2 ведется подсчёт упоминаний слова в документе. Значения складываются в QMap WordsDocument для соответствующего ключа;



Рисунок 3.3 - Структура Words Document

1. Одновременно с пунктами 1 - 3 для объекта класса Document, соответствующего обрабатываемому тексту, задаются поля documentName, documentType, ND;
2. Полностью сформированный со своими полями document <Document> кладётся в documentsVector класса DocumentCollection;
3. Пункты 1 - 6 повторяются для следующего текста из коллекции;
4. Подсчитывается коэффициент ;
5. Одновременно с пунктом 1 - 6 подсчитываются слова в общей коллекции. Считываемые слова в пункте 2 складываются в качестве ключей в QMap WordsCollection, таким образом, чтобы был уникальный набор слов;
6. Одновременно с пунктом 8 подсчитывается частота упоминаемости документа с этим словом в коллекции. Основная идея заключается в том, чтобы получить для каждого слова во всей коллекции коэффициент .

, (3.1)

где - количество документов, в которых слово упоминается хотя бы 1 раз; NDC = 150 - общее число документов в коллекции

Алгоритмически выгодно при обработке очередного текста, если встретилось слово первый раз (последующие пропускаются), то изменение частоты происходит по следующей формуле:

, (3.2)

где .

Таким образом, после обработки всех текстов из коллекции получится WordsCollection.



Рисунок 3.3 - Структура Words Collection

Заполняются все оставшиеся поля DocumentsCollection NT, NDC, NWC и так далее (кроме WordsTrash);

Фильтрация представляет собой удаление общеупотребительных слов по частоте упоминания в коллекции документов. Алгоритм фильтрации достаточно просто в использовании и понимании.

Во-первых, берется WordsCollection из DocumentsCollection. Для каждого слова требуется проверить выполнение следующих условий:

1. Если , то слово остаётся;
2. Если , то слово удаляется;

Одновременно удаляется вся запись из WordsCollection.

Если удаляется, его также необходимо удалить из всех WordsDocument, для всех Document, где оно встречается.

После всех операций необходимо провести пересчёт , поскольку число уникальных и прошедших фильтрацию слов в коллекции (и документах) изменилось. Все выбрасываемые слова требуется поместить в QMap <QString, double> wordsTrash вместе со своими коэффициентами . Данная структура потребуется для анализа общеупотребительных слов. Для первого опыта примем . В дальнейшем будет варьироваться для анализа фильтрации.

Вывод результатов будет осуществляться следующим образом. В директории выходных данных создаётся файл CommonData.txt. В него выводится данные из WordsCollection (класс DocumentCollection) двумя столбцами.

Затем создаётся файл trashWords.txt. В него двумя столбцами выводятся данные из QMap <QString, double> .

Создаются тематические директории с файлами наподобие xxx\_politic\_01.txt и др. В эти файлы последовательно выводятся файлы из WordsDocument соответствующих объектов класса Document из documentsVector.

## 3.3 Программа, реализующая нейронную сеть, NeuralNetClassificator

### 3.3.1 Структура программы NeuralNetClassificator

Для полного понимания требуется ввести несколько условных обозначений, а именно обозначения исходных данных:

1. , где - это номер темы. Максимальное количество тем -6.
2. , где - номер документа для j-ой темы.
3. , где - количество слов для i-ого документа из j-ой темы.

Основной модуль программы обладает следующей функциональностью:

1. Запуск программы;
2. Запуск интерфейсной части;
3. Запуск всех необходимых модулей (методов) для реализации функциональной части программы;
4. Инициализация структур данных;

Программный продукт должен обладать минимальной интерфейсной частью с функциональностью:

1. Кнопка START для запуска расчётной части программы;
2. FileDialog для задания директории с исходными данными;
3. Окна для задания различных коэффициентов: и т.д.;
4. Виджет ProgressBar для отображения расчётной части программы;
5. Виджет FileDialog для вывода результатов в файлы;
6. Стандартные средства работы с окном интерфейсной части: иконка закрытия программы, иконка свертывания/развертывания окна;
7. Окно пошагового прохождения программы с выводом текущих результатов номер нейрона-победителя и так далее;

Разрабатываемый программный продукт должен содержать следующие функциональные модули и обладать следующей структурой:

1. Класс NeuronSet, который содержит функциональность и стуктуру одного нейрона, а именно:

* int NeuronId;
* QString neuronName- задаётся при инициализации;
* QString neuronTopic - определяется в конце обучения;
* QVector clusterDocuments - содержит имена (ссылки) текстов, относящихся к данному нейрону. Обновляется на каждом шаге обучения;
* double results - содержит процесс правильно распознанных текстов на каждом шаге обучения;
* double и другие численные данные относящиеся к данному нейрону;
* QMap <QString, double> hmap - карта коэффициентов j-ого (текущего) нейрона, где ключом являются слова из коллекции;
* используемые методы: void initialisation() - инициализация ( представляет собой метод присвоения начальных весов слов в нейроне, а также присвоение значений характерным переменным, перечисленным выше); double calculateR(QMap<QString,double> &wordsDocument) - расчёт (расчёт расстояний, представляющих собой сумму весов нейрона, совпадающими со словами документа); void trainingWinner(QMap<QString,double> &wordsDocument, double koef1) - обучение победителя (данный метод позволяет увеличивать веса слов, имеющихся в документе); void trainLooser (QMap<QString,double> &wordsDocument, double koef2) - обучение проигравшего (данный метод соответственно уменьшает веса слов, содержащихся в документе);

1. Класс NeuronSet, обеспечивающий функциональность нейронной сети, включает в себя:

* Структуры данных текстов и коллекции из предыдущей программы;
* QVector [Neuron] neurons - содержит 6 объектов класса Neuron;
* Величины и другие коэффициенты, относящиеся к сети в целом;
* Используемые методы:
* void nnetInitialization() - инициализация (вызов инициализации j = 1÷6 Neuron). Данный метод позволяет присвоить начальные веса словам, характерным для текущей темы;
* void nnetAdjustment() - организация основного цикла с вызовом необходимых методов из Neuron (в режиме прогона и пошаговом режиме);
* int calculateDelta(QMap<QString, double> &wordsDocument) - расчет и определение победителя. Победителем является нейрон с наибольшей ;
* void resultsAnalisys() - анализ результатов;

### 3.3.2 Функционирование программы NeuralNetClassificator

В основе работы программы лежит упрощенный метод классификации текстов. Соответственно все величины и переменные, принятые обозначения следует брать из пункта 2.2 данной работы.

Начало работы программы начинается с её запуска. Создаётся окно интерфейсной части. Затем создаются и инициализируются структуры с хранящимися в них данными. После проведенных операций программа считается готовой к работе

Пользователь, используя виджет FileDialog, указывает директорию с исходными данными. В окне задаются необходимые для работы вычислительного алгоритма параметры, такие как , и так далее. Также задаются необходимые текста для инициализации сети.

Алгоритм самообучения сети начинается с проведения инициализации 6 нейронов сети. Затем в цикле (пошагово или прогоном) вычисляются характерные расстояния для каждого текста. После вычисления расстояний выбирается наибольшее , определяющее победителя. Вычисляются Δ.

После проведенных операций, перечисленных выше, производится обучение нейронов (победителей и проигравших). Производится анализ информации об обучении.

Закончив работу с одним документом, можно переходить к следующему, начиная работу цикла заново. Следует помнить, что тексты подавать требуется в произвольном порядке, а не последовательно по темам. Данное условие помогает избежать лишних проблем в виде перетренированности некоторых нейронов.

После прогона всей текстовой коллекции требуется повторять цикл обучения до . При этом по заданным законам должны меняться следующие величины: , . В конце каждого цикла обязательно должна анализироваться информация о ходе обучения, такая как процент распределения текстов по кластерам, , и так далее.

По окончании цикла самообучения необходимо провести окончательный анализ результатов, которые сохраняются в соответствующие файлы outputData.

## 3.4 Заключительные исследования и анализ результатов

### 3.4.1 Анализ выполнения предварительной фильтрации

При подготовке исходных данных к анализу необходимо убрать широкоупотребительные слова из общей текстовой коллекции. Данные слова нехарактерны для тем и поэтому не принадлежат какому-то одному определённому кластеру. От результатов предварительной фильтрации зависят результаты работы алгоритма обучения нейронов и, следовательно, эффективность поиска. Поэтому для получения коллекции слов, с содержащимися в ней характерными для тем словами, необходимо задать порог фильтрации и постепенно его снижать до тех пор, пока слова коллекции не будут удовлетворят требованиям. Поскольку анализ фильтрации будет определяться исключительно визуальным способом, то имеет смысл также просматривать список отфильтрованных слов с соответствующими для них весами.

В ходе анализа отсеянных слов был подобран самый оптимальный коэффициент фильтрации, убирающий нехарактерные для обозначенных тем слова, такие как предлоги, артикли, местоимения и междометия. Величина .

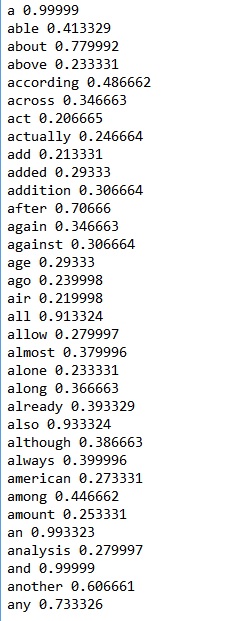


Рисунок 3.4 - Пример широкоупотребительных слов с

В процессе фильтрации происходит подсчёт слов в зависимости от .

Ниже прилагаются примеры широкоупотребительных слов с различными коэффициентами фильтрации.

При , количество слов , отфильтрованные слова - 0%; при , количество слов , отфильтрованные слова - 46,29%; при , количество слов , отфильтрованные слова - 52,44%; при , количество слов , отфильтрованные слова - 61,42%; при , количество слов , отфильтрованные слова - 73,08%.

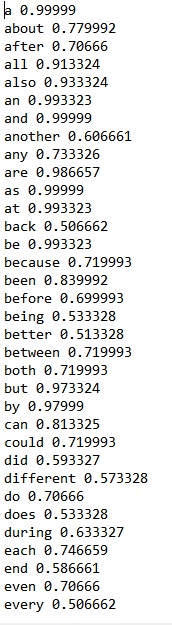


Рисунок 3.5 - Пример широкоупотребительных слов с

Таким образом, различные значения коэффициента предварительной фильтрации влияют на количество и характер отсеиваемых слов. Как правило, проверку не проходят широкоупотребительные слова и, увеличивая или уменьшая порог фильтрации, можно подобрать оптимальное значение.

### 3.4.2 Анализ обучения

В результате обучения нейронов большой вес приобретают слова характерные для темы того или иного нейрона. Следовательно, ключевой задачей становится оценка деятельности работы разработанного алгоритма обучения. В ходе настройки сети, которая заключается в подборе коэффициентов . Данные коэффициенты подбираются исключительно пользователем.

- коэффициент настройки нейрона-победителя согласно методике пункта 2.3.2. В ходе отладки оптимальным значение принято считать

- коэффициент настройки нейрона-проигравшего. Оптимальное значение

- порог фильтрации, который отсеивает широкоупотребительные слова. Оптимальное значение

- пороговое значение, с которым сравнивается .

- количество циклов обучения текстовой коллекции. Оптимальное значение .

Для анализа распознавания текстов вводится коэффициент распознавания , где - количество правильно распознанных текстов; -количество неправильно распознанных текстов. Ниже прилагаются гистограммы зависимости при меняющихся параметрах .

Таким образом, перечень оптимальных значений варьируемых параметров представляет собой: , , , .

На Рисунке 3.7 изображена гистограмма зависимости коэффициента распознавания текстов от коэффициентов , использующихся в вычислительных операциях при тренировке соответственно нейрона-победителя и проигравших нейронов. В процессе выполнения цикла обучения эти коэффициента увеличиваются каждую итерацию. Однако при измене стартовых значений коэффициентов веса слов в нейроне при обучении будут увеличиваться и уменьшаться в большей степени и с большей скоростью.

Рисунок 3.6 - Гистограмма зависимости

Рисунок 3.7 - Гистограмма зависимости

При анализе Рисунка 3.8, на котором изображена гистограмма зависимости коэффициента распознавания от значения . При уменьшении порога нейроны начинаются обучаться и соответственно менять веса слов на более ранних итерациях обучающего цикла и, следовательно, при оптимальных настройках один или несколько нейронов начинают перетренировываться. В результате большинство текстов при распределении сгруппировываются в одном кластере перетренированного нейрона.

Рисунок 3.8 - Гистограмма зависимости

Таким образом, правильное распознавание по всех имеющимся текстам составляет 90,38%. Данная величина была рассчитана при оптимальных настройках, полученных в ходе отладки программы и не допускающих перетренировку нейронов сети.

Вывод: были рассмотрены общие требования к программному обеспечению. Также был проведён анализ структуры и методов двух разработанных программных модулей. Первый программный модуль выполняет функцию подготовки исходных входных данных нейронной сети, в том числе предварительную фильтрацию общеупотребительных слов. Второй программный модуль обеспечивает функциональность нейронной сети, а именно все необходимые вычисления, требуемые для работы алгоритма обучения нейронов. В ходе отладки были подобраны оптимальные значения варьируемых параметров, при которых достигается наибольший коэффициент распознавания текстов.

# 4. БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

При проектировании производственных систем чрезвычайно важно применять эргономические принципы, которые требуются для создания оптимальной рабочей среды персонала. При этом значительно повышается уровень комфорта и профессионального мастерства. Эргономически-правильно спроектированная система обеспечит не только безопасность персоналу, но и улучшит его профессиональные показатели и результативность.

ГОСТ Р ИСО 6385-2007 "Эргономика. Применение эргономических принципов при проектировании производственных систем" рекомендует при проектировании производственного оборудования, аппаратных и программных средств сделать акцент на интеллектуальные, а также физико-технические аспекты, относящиеся к оборудованию[4].

Большое внимание следует уделить ключевым элементам интерфейса, благодаря которому происходит обмен информацией между пользователем и техникой. Интерфейс разработанного программного обеспечения прост и в то же время содержит исключительно адекватную и детальную информацию о параметрах.

Все элементы интерфейса находятся в зоне досягаемости и легко-доступны. Это достигается наличием вкладочной структуры в разработанном программном модуле. Каждая вкладка содержит элементы управления и индикаторы, требуемые для работы соответствующих этапов алгоритма программы. На Рисунке 4.1 изображена первая вкладка программы Input, в которой происходит инициализация входных данных.

Чтобы уменьшить вероятность ошибок персонала, все элементы управления подписаны. Чтобы не вводить персонал в заблуждение и быть совместимыми с характеристиками человеческого восприятия, все элементы выбраны и размещены на интуитивно понятных начинающему пользователю ПК местах, например наличие значков закрытия, свертывания/развертывания программы в правом верхнем углу программного модуля (Рисунок 4.1) и скомпонованных элементов (Рисунок 4.2).

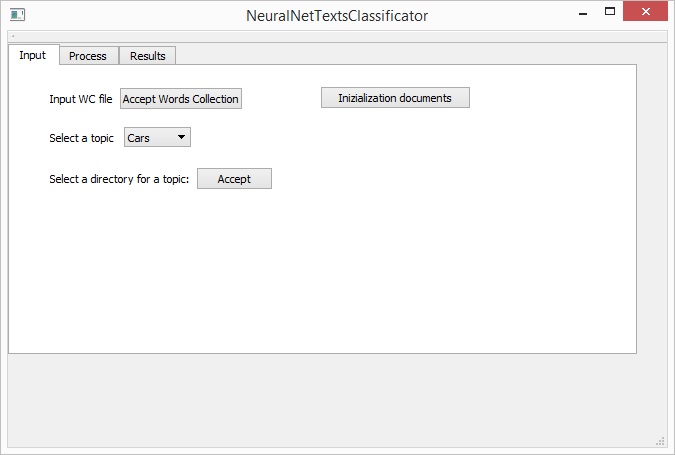


Рисунок 4.1 - Интерфейс спроектированного программного модуля NeuralNetClassificator

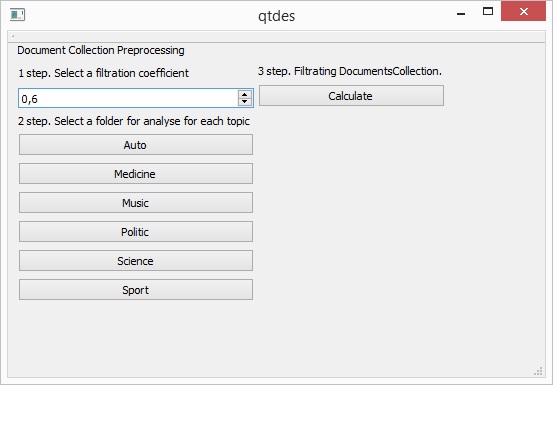


Рисунок 4.2 - Интерфейс программного модуля предварительной фильтрации Qtdes

Информация, получаемая от индикаторов, понятна, однозначно толкуется и не несёт в себе избыточного характера. Шрифт и его размер подобран таким образом, чтобы не вызывать неудобств у пользователя. Элементы управления чётко видны и различимы.

Элементы управления расположены достаточно близко, причём для упрощения работы оператора они расположены в последовательном порядке и скомпонованы по этапам работы алгоритма, как например на Рисунке 4.2, где присутствуют 3 скомпонованных элемента, содержащие присущие им управляющие воздействия.

Вывод:в результате анализа аспектов эргономики разработанного программного обеспечения были выявлены факторы, нарушающие взаимодействие и обмен информацией между технической системой и человеком, а также были предприняты меры по их избеганию. Интерфейс программы был подробно проанализирован на соответствие эргономическим требованиям.

# ЗАКЛЮЧЕНИЕ

В первой главе был проведен теоретический обзор предметов и явлений, используемых в данной работе. Было рассмотрено понятие кластерного анализа и основных методов кластеризации, нейронной сети и её общая структура. Также была оценена релевантность тех или иных методов, а также перечислены их достоинства и недостатки.

Во второй главе были доработаны методы векторного квантования, а также описана их математическая модель. С учётом присущих им свойств был выбран наиболее подходящий для поставленной задачи.

В третьей главе были разработаны программные модули с учётом выбранной методики классификации текстов, такие как: программа подготовки исходных данных, осуществляющая парсинг текстов, запись в соответствующие структуры и предварительную фильтрацию; основной программный модуль, осуществляющий работоспособность нейронной сети, обучение нейронов и анализ распознавания текстов.

В результате работы был разработан оригинальный алгоритм классификации текстов по тематикам с использованием нейронной сети векторного квантования с обучением без учителя. Также была решена задача предварительной фильтрации исходных текстов от общеупотребительных слов. Алгоритмы фильтрации и нейронной сети были реализованы программно с использованием языка программирования С++ под QT. Алгоритмы были успешно протестированы и отлажены. В результате проделанных экспериментов были достигнуты оптимальные параметры настройки алгоритмов, обеспечивающие наилучший результат классификации текстов. Также был разработан интерфейс для двух программных модулей, выполняющих вышеперечисленные операции. Необходимо отметить, что в дальнейшем алгоритмы могут быть доработаны с учётом других методов векторного квантования.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 - Википедия [Электронный ресурс] – URL: http://ru.wikipedia.org/wiki/ (дата обращения: 24.04.2016)

2 - Андреев А.М., Березкин Д.В., Морозов В.В. Метод кластеризации документов текстовых коллекций и синтеза аннотации кластеров. - МГТУ им. Н.Э. Баумана, 2005

3 - Лекционные материалы Вологодского Государственного Технического Университета [Электронный ресурс] - URL: http://zdo.vstu.edu.ru/umk/html/manual/L5\_4.html (дата обращения: 28.04.2016)

4 - Н.В. Блажко, В.А. Буканин, О.В. Демидович и др. Вопросы обеспечения безопасности в выпускных квалификационных работах. - СПбГЭТУ "ЛЭТИ", 2014

# ПРИЛОЖЕНИЕ А

Заголовочные файлы программы предварительной фильтрации

#ifndef QTDES\_H

#define QTDES\_H

#include <Qmap>

#include <string>

#include <algorithm>

#include <cstdlib>

#include <iomanip>

#include <vector>

#include <QApplication>

#include <QtCore/QFile>

#include <QtCore/QTextStream>

#include <iostream>

#include <fstream>

#include <clocale>

#include <QDebug>

#include <QMainWindow>

#include <QVector>

#include <QtWidgets>

#include <QMainWindow>

#include <QtCore/QFile>

#include <QtCore/QTextStream>

#include "QFileDialog"

#include <QString>

#include <QtWidgets>

#include <QDir>

#include <QFile>

#include <QFileInfo>

#include <QTextStream>

namespace Ui {

class qtdes;

}

class Document

{

public:

QString documentName;//имя документа

QString documentType;//тип документа

int ND; //число слов в документе

QMap <QString, double> WordsDocument;//список для хранения слов данного документа(ключ) и числа упоминаний слова в документе

};

class DocumentCollection

{

public:

QMap <QString, double> WordsCollection;//список для хранения слов в коллекции(ключ) и частоты упоминаемости документов в коллекции

QVector <Document> documentsVector;// вектор для хранения объектов класса Document;

int NT=6;//число тем

int NDC=150;//число документов в коллекции

int NWC;//число слов в коллекции

QMap <QString, double> WordsTrash;//отфильтрованные слова

};

class qtdes : public QMainWindow

{

Q\_OBJECT

public:

explicit qtdes(QWidget \*parent = 0);

~*qtdes*();

void obrabotkadirektoryy(QString filenames, QString type);

void vyvodvfaily(QString adresdirektoryy, QString type);

private slots:

void on\_autoButton\_clicked();

void on\_medicineButton\_clicked();

void on\_sportButton\_clicked();

void on\_musicButton\_clicked();

void on\_scienceButton\_clicked();

void on\_politicButton\_clicked();

void on\_pushButton\_2\_clicked();

void on\_calculateButton\_clicked();

void on\_KfiltrSpinBox\_valueChanged(double arg1);

void on\_acceptButton\_clicked();

void on\_KfiltrSpinBox\_editingFinished();

private:

Ui::qtdes \*ui;

};

# ПРИЛОЖЕНИЕ Б

**Заголовочные файлы программы обучения нейронов. Заголовочный файл document.h**

#include<QtCore>

class Document

{

public:

Document();

~Document();

QString getDocumentName() const;

void setDocumentName(const QString &value);

QString getDefaultDocumentType() const;

void setDefaultDocumentType(const QString &value);

QString getFinalDocumentType() const;

void setFinalDocumentType(QString &value);

QMap<QString, double> getWordsDocument() const;

void setWordsDocument(const QMap<QString, double> &value);

bool checkParameters();//check initial pframeters of the document; return true if OK

void controlOutput();

private:

QString documentName;//The name of the document

QString defaultDocumentType;//The default topic of the document

QString finalDocumentType;//The final topic of the document

QMap<QString, double> wordsDocument;// The collection of word and its

//coeff. for current documment

};

**Заголовочный файл documentscollection.h**

#include <document.h>

class DocumentsCollection

{

public:

DocumentsCollection();

enum topicsList{ cars=0, medicine=1, music=2, politic=3, science=4, sport=5};

//enum topicsList{ cars=0, music=1, politic=2, science=3, sport=4};

QString getTopic(int value) const;

int getNT() const;

void setNT(int value);

int getNDC() const;

void setNDC(int value);

int getNWC() const;

void setNWC(int value);

QVector<Document> getDocumentsVector() const;

void setDocumentsVector(const QVector<Document> &value);

QMap<QString, Document> getInitializationDocuments() const;

void setInitializationDocuments(const QMap<QString, Document> &value);

bool checkParameters();//check initial pframeters of documentsCollection

QMap<QString, double> getWordsCollection() const;

void setWordsCollection(const QMap<QString, double> &value);

void controlOutput();//control output for all document collection data;

private:

int NT;//number of topics

int NDC;//number of documents for collection

int NWC;//number of words for all collection

QVector<Document> documentsVector;//Vector for document collection

QMap<QString,Document> initializationDocuments;//documents for

//initialization of neuralNet

QMap<QString,double> wordsCollection;//Map for all words of all

//documents in the collection and its frequencies

};

**Заголовочный файл neuralnettextclassificator.h**

#include <documentscollection.h>

#include <document.h>

#include <neuron.h>

#include <QMainWindow>

#include <QDir>

#include <QFileDialog>

#include <neuronset.h>

#include <clocale>

#include <QWidget>

#include <Qmap>

#include <string>

#include <algorithm>

#include <QDebug>

#include <QTextStream>

namespace Ui {

class NeuralNetTextsClassificator;

}

class NeuralNetTextsClassificator : public QMainWindow

{

Q\_OBJECT

public:

explicit NeuralNetTextsClassificator(QWidget \*parent = 0);

~*NeuralNetTextsClassificator*();

private slots:

void on\_acceptWCbutton\_clicked();

void on\_acceptdirectorybutton\_clicked();

void on\_inizializationdocumentsbutton\_clicked();

QMap <QString, double> init (const char\* fname);

void on\_topicselect\_activated(const QString &arg1);

void on\_startlearning\_button\_clicked();

void on\_progressBar\_valueChanged(int value);

void on\_resultbutton\_clicked();

private:

Ui::NeuralNetTextsClassificator \*ui;

QDir currentDir;

QStringList filesDocument;

QString fileDocumentCollection;

};

**Заголовочный файл neuron.h**

#include <Qtcore>

#include "documentscollection.h"

using namespace Qt;

class Neuron

{

public:

explicit Neuron(void);

explicit Neuron(QMap<QString,double>& wordsCollection,const QString &neuronName,

const QString &neuronDefaultTopic, int& neuronId);

virtual ~*Neuron*(void);

QString getNeuronName() const;

void setNeuronName(const QString &value);

QString getNeuronDefaultTopic() const;

void setNeuronDefaultTopic(const QString &value);

QString getNeuronFinalTopic() const;

void setNeuronFinalTopic(const QString &value);

int getNeuronId() const;

void setNeuronId(int value);

int getNWC() const;

void setNWC(int value);

QMap<QString, double> \*getHmap() const;

void setHmap(QMap<QString, double> \*value);

double getWeightByWord(QString& word)const;

void setWeightByWord(QString& word, double h);

double getR() const;

void setR(double value);

double getDelta() const;

void setDelta(double value);

bool isWinner() const;

void setWinner(bool value);

void initialisation();

double calculateR(QMap<QString,double> &wordsDocument);

void trainingWinner(QMap<QString,double> &wordsDocument, double koef1);

void trainLooser (QMap<QString,double> &wordsDocument, double koef2);

void controlOutput(bool value);

private:

QString neuronName; // The Name of current neuron

QString neuronDefaultTopic; //The default (initial) topic of the neuron;

QString neuronFinalTopic; // The final topic of the neuron

int neuronId; //The id of current neuron

int NWC; //inputs number (words number in the collection)

double r;//the estimation of distance between current neuron and document; r= sum(h);

double delta;//winner price- the estimation of degree of winner; delta= r/sum(r);

bool winner;//true if neuron is winner

QMap<QString,double> \*hmap;//The map of inputs weights, where <QString>word is a key and <double>h

};

**Заголовочный файл neuronset.h**

#include <neuron.h>

class NeuronSet

{

public:

NeuronSet();

~NeuronSet(){};

double getK1() const;

void setK1(double value);

double getK2() const;

void setK2(double value);

double getDeltaThreshold() const;

void setDeltaThreshold(double value);

int getTmax() const;

void setTmax(int value);

void mainProcess();//the main process of NeuralNetwork lifecycle;

void controlOutput(bool value);//control output for all neurons set data;

void nnetInitialization();// neural net initialization;

bool checkInputParameters();// check input paramrters: return true if OK;

void nnetAdjustment();// neural network adjustment;

void resultsAnalisys();// results analisys;

int calculateDelta(QMap<QString, double> &wordsDocument);//calculate delta for all neurons

QVector<Neuron> getNeurons() const;

void setNeurons(const QVector<Neuron> &value);

DocumentsCollection getDc() const;

void setDc(const DocumentsCollection &value);

private:

double k1;//adjustment koef. for winner neuron;

double k2;//adjustment koef. for looser neuron;

double deltaThreshold;//threshold value for delta;

int tmax;//max time of adjustment;

QVector<Neuron> neurons;//set of neuron;

DocumentsCollection dc;//document collection object (singleton!);

};