МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра технологии программирования

Херндлхофер Никита

Система выделения основной смысловой части текста

Дипломная работа

студента 4 курса 8 группы

Руководитель:

Д.Т.Н., профессор Курбацкий
Александр Николаевич

Заведующий кафедрой ТП

_____

Минск, 2019

АННОТАЦИЯ

Херндлхофер Н. Анализ алгоритмов обработки естественного языка и распознавания речи: Курсовой проект / Минск: БГУ, 2018.

В дипломной работе рассмотрен вопрос изучения алгоритмов распознавания речи и обработки естественного языка. Для демонстрации разработаны приложения.

ANNOTATION

Herndlofer N. Analysis of algorithms for speech recognition and natural language processing: Course project / Minsk: BSU, 2018.

In the diploma work, the study of algorithms for speech recognition and natural language processing is considered. Developed applications for demonstration.

АНАТАЦЫЯ

Херндлхофер Н. Аналіз алгарытмаў распазнавання прамовы і апрацоўкі натуральнай мовы: Курсавы праект / Мінск: БДУ, 2018.

У дыпломнай працы разгледжаны пытанне вывучэння алгарытмаў распазнавання прамовы і апрацоўкі натуральнага мовы. Для дэманстрацыі распрацаваны прыкладання.

Реферат

Ключевые слова: РЕЧЬ, ЯЗЫК, ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА, РАСПОЗНАВАНИЕ РЕЧИ, СЛОВО, ДОКУМЕНТ, ТЕКСТ

Объект исследования: Анализ современных алгоритмов обработки естественного языка, разработка и анализ работы выбранного алгоритма.

Цель работы: Описать подходы для обработки естественного языка. Провести анализ современных алгоритмов для решения проблемы компьютерного анализа естественных языков. Разработать приложение, демонстрирующее алгоритм обработки естественного языка.

Методы исследования – теория программирования, анализ.

Область применения: сфера машинного обучения, бизнес-сфера.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Распознавание речи и анализ текста – одни из самых распространенных задач обработки естественного языка. Распознавание речи используется на данный момент в основном в 4 отраслях:

- Голосовое управление;
- Голосовые команды;
- Голосовой ввод текста;
- Голосовой поиск.

Преимуществом всех систем, использующих распознавание голоса для ввода данных всегда являлось большая дружественность к пользователю, нежели в системах, использующих текстовый ввод. Достаточно успешными примерами таких систем можно назвать сейчас голосовые ассистенты, такие как Google Now, Apple Siri, Amazon Alexa, Microsoft Cortana и другие.

Анализ текста, в свою очередь, в последнее время привлекает все больше внимания в различных областях и сферах деятельности, таких как безопасность информации, коммерция и наука.

Например, многие пакеты анализа текста нацелены на рынок приложений безопасности, например, на анализ текста из новостных сайтов.

Исследования и разработки крупнейших IT-компаний исследуют технологии анализа текста с целью автоматизации процессов анализа и извлечения данных.

В данной дипломной работе рассмотрены некоторые алгоритмы обработки естественного языка и распознавания речи.

Был реализован отдельно выбранный алгоритм анализа текста и проанализирован его потенциал.

ГЛАВА 1. ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА

1.1. Что это такое?

Обработка естественного языка – это общее направление искусственного интеллекта и математической лингвистики. Оно изучает проблемы компьютерного анализа и синтеза естественных языков. Решение этих проблем означает создание более удобной формы взаимодействия человека и компьютера.

1.2. Зачем это нужно?

Обработка естественного языка отлично вписывается в концепцию построения естественно-языкового пользовательского интерфейса. Однако не только пользовательский интерфейс может быть целью для изучения обработки естественного языка. Помимо этого, такие технологии могут уже имеют широкое распространение в различных сферах жизни:

- Телефония: автоматизация обработки входящих и исходящих звонков при помощи создания голосовых систем самообслуживания в частности для: проведения опросов, анкетирования, сбора информации, информирования, получения справочной информации и консультирования, изменения параметров действующих услуг, заказа услуг/товаров и любых других сценариев.
- Решения "Умный дом": интерфейс управления системами «Умный дом».
- Бытовая техника и роботы: интерфейс электронных роботов; голосовое управление бытовой техникой и т.д.
- Десктопы и ноутбуки: голосовой ввод в компьютерных играх и приложениях.
- Автомобили: управление системой навигации, радио, звонками и прочим в салоне автомобиля не отвлекаясь от дороги.
- Сервисы, направленные на облегчение жизни людям с огранниченными возможностями.
  И этот список можно постоянно пополнять.

1.3. Основные сложности

Качество понимания и распознавания компьютером речи и текста зависит от большого количества факторов, таких как: сам язык, самого собеседника, и т.д. Для большего понимания приведу несколько примеров:

- Порядок слов в предложении имеет большое значения для понимания текста, к примеру: «Бытие определяет сознание» - что определяется из чего?

- Во многих языках также существует большая сложность использования знаков препинания, развитая морфология и большое количество служебных слов, что также затрудняет понимание программой речи.

- Региональные диалекты, сленг и неологизмы затрудняют разработку, так как редко документируются.

- Также возникает проблема с произношением и омонимами (- это слова одинаковые по написанию или звучанию, но разные по значению морфемы и другие единицы языка), например слово «норка» должно трактоваться в зависимости от контекста как животное или как тоннель под землей.

- И количество таких проблем постоянно растет, так как человеческие языки постоянно развиваются.

1.4. Главные задачи обработки естественного языка

- Распознавание речи – процесс преобразования речи в цифровую информацию, с которой возможно работать на компьютере.

- Анализ текста – декомпозиция текста для получения информации из коллекций документов.

- генерация текста – процесс создания корректного и осмысленного текста автоматически при помощи программного обеспечения.

- Синтез речи – создание речи по заданному тексту.

Глава 2. Распознавание речи

2.1 Преобразование аналогового сигнала в цифровой

Что такое речь? Речь – это последовательность звуков, а звук, в свою очередь, - это звуковые волны различных частот. Волна же, как известно, имеет амплитуду и частоту. Для того, чтобы сохранить аналоговый звуковой сигнал на цифровом носителе его нужно разбить на отрезки и создать массив значений состоящий из усредненного значения на каждом отрезке. В этом и есть основное отличие аналоговых сигналов от цифровых – представляет из себя функцию, в то время как цифровой состоит из отдельных координат, как это показано в рисунке 1:

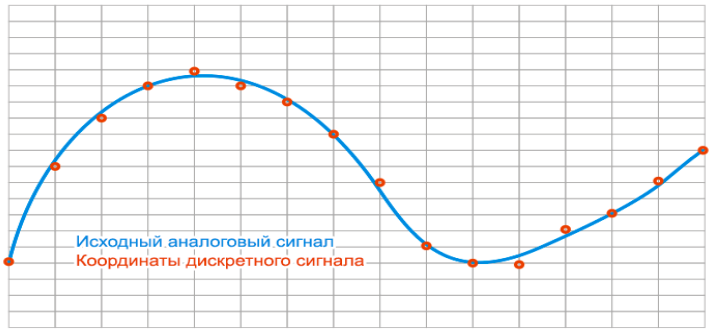

рис. 1

И в итоге, можно получить ступенчатую функцию, которая будет иметь вид показанный на рисунке 2:

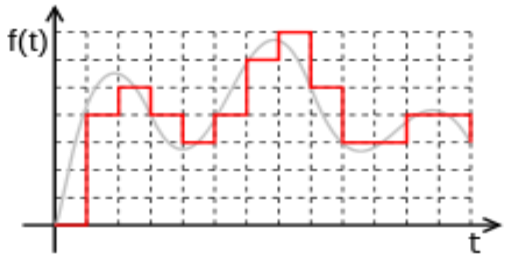

рис. 2

2.2 Сохранение и представление в памяти компьютера

Каждый формат звуковых файлов по-разному представляет звук в памяти, я буду рассматривать один из самых простых – WAV файл (рис. 3).

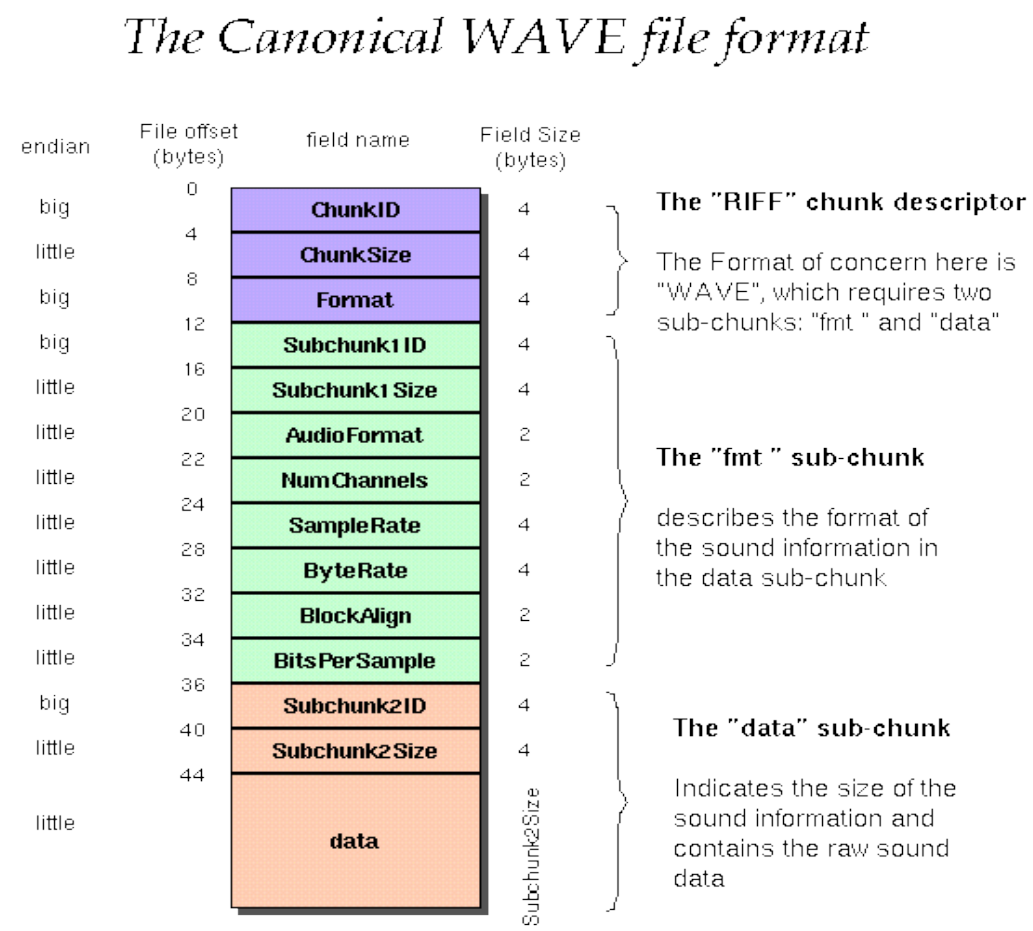

рис. 3

Этот формат подразумевает наличие двух блоков в каждом файле. Первый блок – заголовок с информацией об аудиопотоке: частоте, битрейте, длине файла и т.д. Представление в коде (C++) такого заголовка будет иметь вид, показанный на рисунке 4.

```
struct WavHeader {
    char              riff[4];        // RIFF Header
    unsigned long     chunkSize;      // RIFF Chunk Size
    char              wave[4];        // WAVE Header

    char              fmt[4];         // FMT header
    unsigned long     subchunk1Size;  // Size of the fmt chunk
    unsigned short    audioFormat;    // Audio format 1=PCM (Other formats are unsupported)
    unsigned short    numOfChan;      // Number of channels 1=Mono, 2=Stereo
    unsigned long     samplesPerSec;  // Sampling Frequency in Hz
    unsigned long     bytesPerSec;    // bytes per second
    unsigned short    blockAlign;     // 2=16-bit mono, 4=16-bit stereo
    unsigned short    bitsPerSample;  // Number of bits per sample

    // The data below depends on audioFormat, but we work only with PCM cases
    char              data[4];        // DATA header
    unsigned long     subchunk2Size;  // Sampled data length
};
```

рис.4

Второй блок состоит из так называемых «сырых» данных – цифрового звукового сигнала, состоящего из амплитуд и частот.

Логика чтения данных в этом случае довольно проста:
Считываем заголовок, проверяем некоторые ограничения (отсутствие сжатия, например), сохраняем данные в специально выделенный массив.

2.3 Обработка данных

Разделение на слова

Теоретически, уже можно начать работать с имеющимися данными, сравнить уже имеющийся образец с каким-нибудь другим, текст которого уже известен, но такой подход не будет устойчив к изменению тембра голоса человека, например, к изменению громкости и скорости произношения. Поэлементным сравнением двух аудиосигналов этого, естественно, добиться нельзя.

Нельзя обрабатывать весь набор входных данных целиком и сразу, вначале его следует разбить по маленьким промежуткам времени на т.н. фреймы. Причем, фреймы должны идти не строго друг за другом, а пересекаться в конце и в начале.

Фреймы более удобны как еденица анализа данных, потому что анализировать волны гораздо удобнее на некотором промежутке, нежели в конкретных точках, а такое расположение, когда фреймы пересекаются, позволяет сгладить результаты их анализа.
Для начала нужно определить какие фреймы или наборы фреймов являются слова. В речи каждое слово всегда отделяется небольшой паузой. Так что можно считать, что каждый непрерывный промежуток звучания является словом. Но ни на одной записи никогда нет абсолютной тишины. Должен существовать порог ниже которого звук является тишиной, а выше – словом. Такой порог называется энтропия. Я буду использовать анализ энтропии громкости в рамках заданного фрейма. Энтропия – мера неопределенности (неупорядоченности) какой-либо системы, например какого-либо опыта, в результате которого могут быть разные исходы. Для того, чтобы рассчитать значение энтропии предположим, что наш сигнал пронормирован и все его значения лежат в диапазоне [-1;1] и построим гистограмму (плотность распределения) значений сигнала фрейма. В итоге формула получится такая:

$$E = \sum_{i=0}^{N-1} P[i] * \log_2(P[i])$$

Итак, значение энтропии получено. Но иногда посередине некоторых слов, на гласных, например, энтропия может становиться меньше, так как громкость звучания уменьшается. Для борьбы с этим нужно ввести понятия «минимального расстояния между словами» и объеденять фреймы, разделенные из-за такого уменьшения. Другая проблема это резкое повышение энтропии из-за посторонних шумов. Эта проблема легко решается при помощи

введения минимальной длины слова. Таким образом наборы меньше минимальной длины отбрасываются.

Преобразование набора фреймов, отвечающих за одно слово

Итак, после предыдущего этапа можно получить массив наборов фреймов, каждый из которых будет словом. Теперь нужно из цифрового сигнала фреймов получить векторное его представление. В науке анализа звука есть так называемый анализ Мел-кепстральных коэффициентов. Мел – еденица высоты звука, основанная на восприятии этого звука нашими органами слуха. И, воспринимаемая человеческим слухом высота звука не совсем линейно зависит от его частоты, что видно на графике (рис. 5):

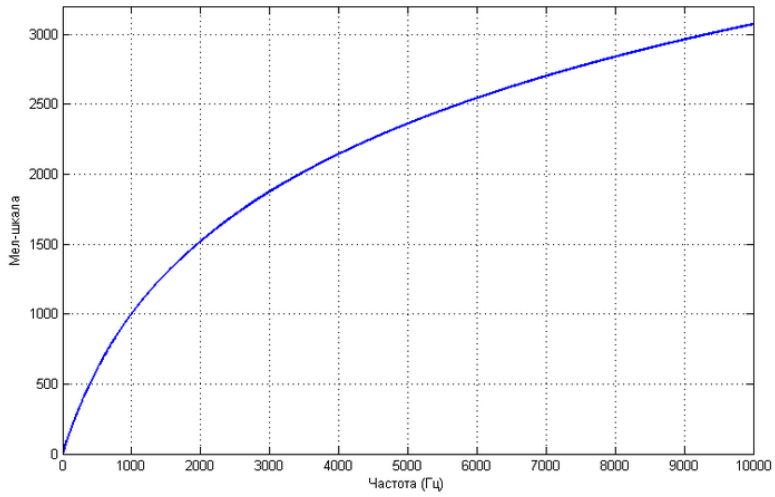

рис. 4

Такая зависимость не претендует на большую точность, но зато описывается простой формулой: $m = 1125\ln(1 + f/700)$
Подобные еденицы изменения лидируют в системах распознавания речи, потому что они помогают приблизиться к человеческому восприятию звука.
Итак, рассчитаем спектр сигнала с помощью дискретного преобразования Фурье.

$$X[k] = \sum_{n=0}^{N-1} x[n] * e^{-2*\pi*i*k*n/N}, 0 \leq k < N$$

Также, к полученным значениям применяется оконная функция Хэмминга, для того чтобы сделать значения более точными и менее зависимыми от тембра голоса человека.

$$H[k] = 0.54 - 0.46 * \cos(2 * \pi * k/(N-1))$$

То есть в результате получим вектор такого вида:

$$X[k] = X[k] * H[k], 0 \leq k < N$$

После этого преобразования по оси X мы имеем частоту (hz) сигнала, а по оси Y — магнитуду.

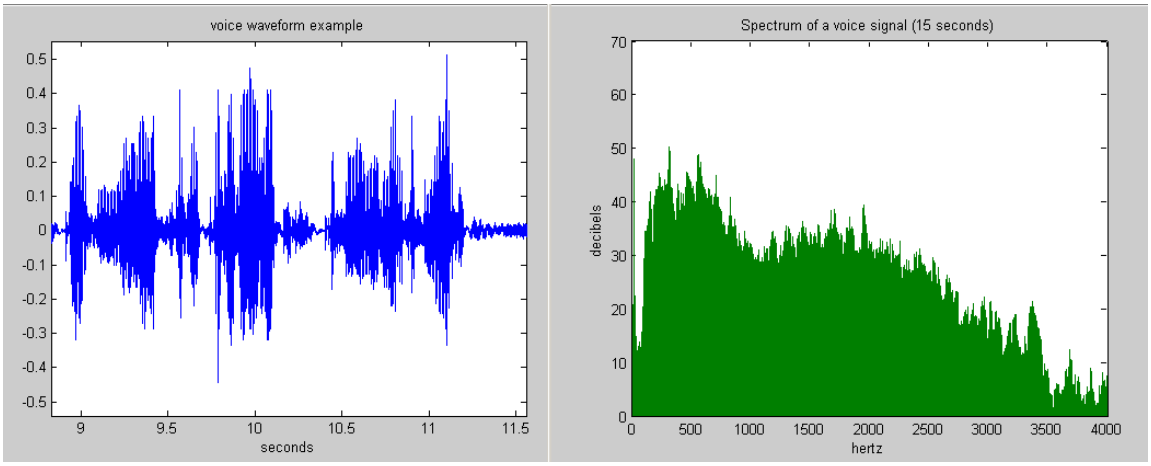

рис. 5

Теперь нужно преобразовать частоту в mel, по формуле, которая была указана выше. Теперь нужно каждый фрейм разложить на некоторое количество мел-коэффициентов, от количества которых будет зависеть точность преобразования, допустим, в нашем случае – 8. Для такого разложения необходимо построить гребенку фильтров, а для этого нужно знать необходимый диапазон частот. Каждый такой фильтр будет являться треугольной оконной функцией.

Предположим, что человеческая речь расположена в диапазоне [260; 9100]hz. Такие фильтры будет иметь такой вид:

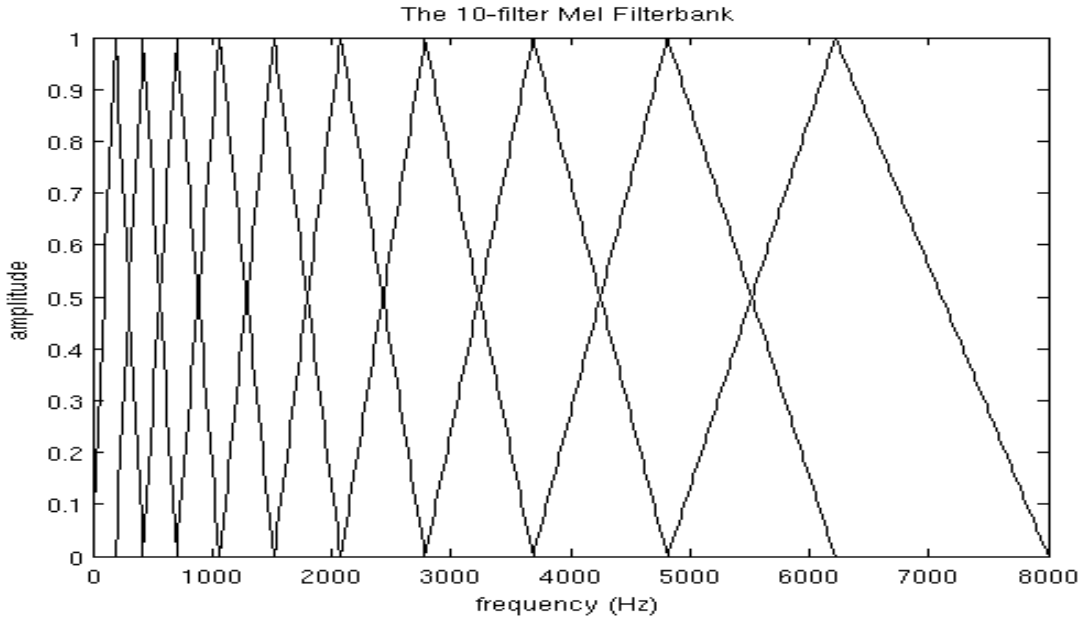

рис. 6

Основание треугольного фильтра прямо пропорционально мел-кепстральному коэффициенту. Это происходит из-за того, что разбиение интересующего нас диапазона частот человеческой речи на обрабатываемые фильтрами диапазоны происходит на шкале mel-ов. Из графика видно, что

чтобы построить 10 треугольных фильтро необходимо 12 опорных точек.

Эти точки будут иметь следующие координаты: [441.25, 122.50, 423.75, 4215.00, 3216.25, 12347.50, 4428.74, 2349.99, 5321.24, 0822.49, 8253.74, 2296.99]

При помощи формулы, обратной формуле перевода hz в mel, переведем эти mel-координаты в hz. Получим следующие значения - [300, 517.33, 781.90, 1103.97, 1496.04, 1973.32, 2554.33, 3261.62, 4122.63, 5170.76, 6446.70, 8000]

Теперь нужно наложить полученную шкалу на спектр фрейма, для которого эта гребенка фильтров была построена. Решив пропорцию можно вывести формулу

f(i) = floor((frameSize+1) * h(i) / sampleRate)

В нашем случае получим на выход значения [4, 8, 12, 17, 23, 31, 40, 52, 66, 82, 103, 128]

Эти значения и будут опорными точками на оси X спектра. По ним можно построить необходимые фильтры используя формулу:

$$H_m(k) = \begin{cases} 0 & k < f(m-1) \\ \dfrac{k - f(m-1)}{f(m) - f(m-1)} & f(m-1) \le k \le f(m) \\ \dfrac{f(m+1) - k}{f(m+1) - f(m)} & f(m) \le k \le f(m+1) \\ 0 & k > f(m+1) \end{cases}$$

рис. 7

Теперь, применение фильтра заключается в попарном перемножении его значений со значениями энергии спектра. Результатом такой операции будет мел-коэффициент. Чтобы снизить чувствительность коэффициента к шумам, можно еще прологарифмировать полученный результат.

$$S[m] = \log(\sum_{k=0}^{N-1} |X[k]|^2 * H_m[k]), 0 \le m < M$$

Теперь, для увеличения значимости первых коэффициентов, можно применить косинусное преобразование:

$$C[l] = \sum_{m=0}^{M-1} S[m] * \cos(\pi * l * (m + \frac{1}{2})/M)), 0 \le l < M$$

Теперь для каждого фрейма мы имеем набор из M mfcc-коэффициентов, которые могут быть использованы для дальнейшего анализа.

Алгоритм распознавания полученного значения

На данный момент, широкое распространение имеет распознавание слов из

некоторого словаря. Именно этот метод я и буду рассматривать. Теперь задача сводится к подбору наиболее похожей модели для данного набора из уже имеющейся базы данных с парами «набор mfcc-коэффициентов» - «текстовое слово»

ГЛАВА 2. АНАЛИЗ ТЕКСТА

Рассмотрим один из алгоритмов анализа текста, а именно – латентно-семантический анализ. Латентно-семантическийй анализ отображает документы и отдельные слова в так называемое «семантическое пространство», в котором и производится выделение ключевых аспектов текста. При этом допускаются следующие предположения:

- Документы представляют собой набор слов, при этом порядок слов значения не имеет
- Значение документов определяется словами, которые используются часто друг с другом, например математическая статья может иметь семантическое значение сопутствующее словам «уравнение», «значение» и «вычисления»
- Каждое слово имеет единственное значение. Это, безусловно, сильное упрощение, но именно оно делает проблему разрешимой.

2.1. Латентно-семантический анализ

Первым делом, из изучаемого текста удаляются все конструкции, слова и так далее, не несущие смысловой нагрузки – это, прежде всего, все союзы, все вводные слова, частицы, предлоги и множество других слов. Часто они определяются при помощи заранее заданных массивов слов и словосочетаний.

Далее производится операция стемминга – выделение основ слов. Для этого используем алгоритм Портера.

Существует достаточно много алгоритмов для стемминга, например как:

а) **Алгоритмы поиска**

Такой стеммер ищет заданную словоформу в таблице поиска. Преимущества этого подхода заключается в его простоте, скорости, а также легкости обработки исключений. Большим недостатком является то, что все флективные формы должны быть явно перечислены в поисковой таблице: новые

или незнакомые слова не будут обрабатываться, даже если они являются правильными (например, таблица ~ таблиц), а также проблемой является то, что таблица поиска может быть очень большой. Для языков с простой морфологией наподобие английского размеры таблиц небольшие, но для сильно флективных языков (например, турецкого) таблица может иметь сотни возможных флективных форм для каждого корня.

b) Алгоритмы усечения окончаний

Существуют алгоритмы, удаляющие окончания из слов для получения словоморф, к примеру «Если слово оканчивается на «ая» - удалить «ая»». Как правило, они используются вместе с другими стеммерами.

c) Аффикс-стеммеры

Существуют алгоритмы обрезающие префиксы и суффиксы слов, например слово предсказание содержит аффиксы «пред» и «н», которые могут быть обрезаны. Как правило, они используются вместе с другими стеммерами.

d) Алгоритмы лемматизации

Более сложным подходом к решению проблемы определения основы слова является лемматизация. Чтобы понять, как работает лемматизация, нужно знать, как создаются различные формы слова. Большинство слов изменяется, когда они используются в различных грамматических формах. Конец слова заменяется на грамматическое окончание, и это приводит к новой форме исходного слова. Лемматизация выполняет обратное преобразование: заменяет грамматическое окончание суффиксом или окончанием начальной формы.

e) Стохастические алгоритмы

Стохастические алгоритмы связаны с вероятностным определением корневой формы слова. Данные алгоритмы строят вероятностную модель и обучаются с помощью таблицы соответствия корневых и флективных форм. Эта модель обычно представлена в виде сложных лингвистических правил, аналогичных по своему характеру правилам, использующимся в алгоритмах усечения окончаний и лемматизации. Стемминг выполняется посредством ввода измененных форм для обучения модели и генерацией корневой формы в соответствии с внутренним набором правил модели, за исключением того, что решения, связанные с применением наиболее соответствующего правила или последовательности правил, а также выбором основы слова, применяются на

основании того, что результирующее верное слово будет иметь самую высокую вероятность (неверные слова имеют наименьшую вероятность).

f) Статистические алгоритмы

Существует большое количество статистических алгоритмов, и многие из них статистически анализирует большое количество текстов, чтобы составить математическую модель, по которой и будет проводиться стемминг.

В реализации моей системы я использовал гибридный стеммер Портера, который использует отсечение окончаний и аффиксов. (Приложение А)

Далее исключаются слова, встречающиеся в единственном экземпляре. Это не сильно влияет на конечный результат, но сильно упрощает математические вычисления.

Для примера возьмем несколько заголовков различных новостных статей, жирным отмечены оставшиеся в итоге слова:

1. Американская **полици**я арестовала **основател**я **WikiLeaks**
2. В **суд**е **США** стартовал суд **прот**ив иранца, который состоит в ИГИЛ
3. **Церемон**ию **вручени**я **Нобелевск**ой **прем**ии мира бойкотируют 19 **стран**
4. В **Великобритан**ии **арестова**н **основател**ь веб-системы **Wikileaks** Джулиан Ассандж
5. ЮАР не принимает **церемон**ию **вручени**я **Нобелевск**ой **прем**ии
6. Американский **суд** выставил приговор **основател**ю **Wikileaks**
7. НАТО и **США** разработали планы обороны **стран** Балтии **прот**ив России
8. **Полици**я **Великобритан**ии выявила местоположение **основател**я **WikiLeaks**, но не **арестова**ла его
9. В Амстердаме и Вене послезавтра будет **вручени**е **Нобелевск**их **прем**ий

На первом шаге требуется составить частотную матрицу индексируемых слов. В этой матрице строки соответствуют индексированным словам, а столбцы — документам. В каждой ячейке матрицы указано какое количество раз слово встречается в соответствующем документе.

	T1	T2	T3	T4	T5	T6	T7	T8	T9
wikileaks	1	0	0	1	0	1	0	1	0
арестова	0	0	0	1	0	0	0	1	0
великобритан	0	0	0	1	0	0	0	1	0
вручен	0	0	1	0	1	0	0	0	1
нобелевск	0	0	1	0	1	0	0	0	1
основател	1	0	0	1	0	1	0	1	0
полиц	1	0	0	0	0	0	0	1	0
прем	0	0	1	0	1	0	0	0	1
прот	0	1	0	0	0	0	1	0	0
стран	0	0	1	0	0	0	1	0	0
суд	0	1	0	0	0	1	0	0	0
сша	0	1	0	0	0	0	1	0	0
церемон	0	0	1	0	1	0	0	0	0

Рисунок 8

0Далее нужно разложить матрицу на 3 составные части так называемым сингулярным разложением. Так матрицу M мы представляем в виде:

$$M = U * W * V^t$$

где U и V^t – ортогональные матрицы, а W – диагональная матрица. Причем диагональные элементы матрицы W упорядочены в порядке убывания. Диагональные элементы матрицы W называются сингулярными числами.

wikileaks	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	-0.64
арестова	0.34	-0	0.07	0.41	-0.42	-0.02	0.1	0.17	0.01
великобритан	0.34	-0	0.07	0.41	-0.42	-0.02	0.1	0.17	-0.01
вручен	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.07
нобелевск	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	0.32
основател	0.57	-0.01	0.01	-0.2	0.13	0.16	-0.16	-0.25	0.64
полиц	0.31	-0	0.05	0.07	0.57	-0.6	0.29	0.37	-0
прем	0	0.52	0.07	-0.06	-0.08	-0.15	-0.17	0.02	-0.25
прот	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
стран	0.01	0.22	-0.31	0.39	0.41	0.56	-0.22	0.4	-0
суд	0.12	0.01	-0.38	-0.62	-0.3	0.12	0.21	0.55	-0
сша	0.02	0.03	-0.61	0.13	-0.05	-0.22	0	-0.25	0
церемон	0	0.38	0.03	0.02	0.08	0.31	0.82	-0.29	0

·

3.41	0	0	0	0	0	0	0	0
0	3.3	0	0	0	0	0	0	0
0	0	2.27	0	0	0	0	0	0
0	0	0	1.49	0	0	0	0	0
0	0	0	0	1.19	0	0	0	0
0	0	0	0	0	0.98	0	0	0
0	0	0	0	0	0	0.71	0	0
0	0	0	0	0	0	0	0.43	0
0	0	0	0	0	0	0	0	0

·

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
-0	0.02	0.65	-0.01	0.59	-0	0.09	-0.01	0.47
0.03	-0.7	-0.04	0.06	0.1	-0.16	-0.67	0.09	0.09
-0.22	-0.24	0.15	0.28	-0.11	-0.68	0.44	0.33	-0.13
0.69	-0.32	0.22	-0.49	-0.12	-0.03	0.27	-0.02	-0.19
-0.27	-0.34	0.44	0.29	-0.13	0.45	0.12	-0.31	-0.45
-0.03	0.3	0.14	-0.17	0.44	-0.15	-0.3	0.24	-0.71
-0.3	0.12	0.4	-0.39	-0.53	0.12	-0.23	0.46	0.13
0.35	0.35	0.35	0.35	-0.35	-0.35	-0.35	-0.35	0

Рисунок 9

Такое разложение позволит очистить шумы во входных данных. Строки, которые соотвествуют меньшим сингулярным значениям дают наименьший вклад в итоговой произведение, таким образом можно отбросить, например, последние столбцы матрицы U и последние строки матрицы V^t, оставив только первые 2. При этом гарантируется оптимальность полученного произведения.

Такое разложение называют двумерным сингулярным разложением:

wikileaks	0.57	-0.01
арестова	0.34	-0
великобритан	0.34	-0
вручен	0	0.52
нобелевск	0	0.52
основател	0.57	-0.01
полиц	0.31	-0
прем	0	0.52
прот	0.02	0.03
стран	0.01	0.22
суд	0.12	0.01
сша	0.02	0.03
церемон	0	0.38

*

3.41	0
0	3.3

*

T1	T2	T3	T4	T5	T6	T7	T8	T9
0.43	0.05	0.01	0.54	0	0.37	0.01	0.63	0
-0	0.02	0.65	-0.01	0.59	-0	0.09	-0.01	0.47

Рисунок 10

Теперь можно составить график точек, соответсвующих отдельным словам или текстам и получится так:

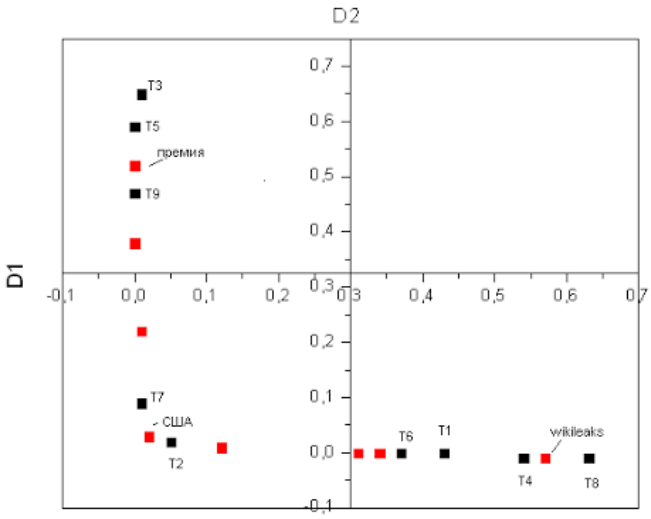

Рисунок 11

Из данного графика видно, что статьи образуют три независимые группы, первая группа статей располагается рядом со словом «wikileaks», и действительно, если мы посмотрим названия этих статей становится понятно, что они имеют отношение к wikileaks. Другая группа статей образуется вокруг слова «премия», и действительно в них идет обсуждение нобелевской премии.

На практике, конечно, количество групп будет намного больше, пространство будет не двумерным а многомерным, но сама идея остается той же. Мы можем определять местоположения слов и статей в нашем пространстве и использовать эту информацию для, например, определения тематики статьи.

2.2. Вероятностный латентно-семантический анализ

Обычный латентно-семантический анализ хорош для знакомства с алгоритмами обработки текста, но он эффективнее всего используется на текстах, в которых слова распределяются Гауссу, с функцией плотности вероятности вида:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu^2)}{2\sigma^2}}$$

где μ - математическое ожидание значения x, а σ – среднеквадратическое отклонение (σ^2 – дисперсия).

К примеру, по нормальному распределению строится графики вероятностей выпадения комбинации чисел на 2 6-гранных кубиках (рисунок 5):

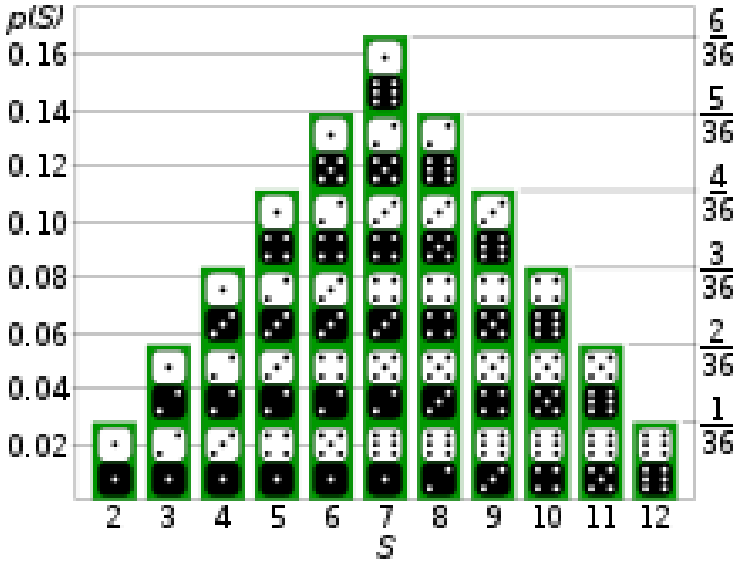

Рисунок 5

Однако в обработке естественного языка существует так называемый закон Ципфа, который показывает, что закономерность распределения частоты слов естественного языка имеет другой график, а именно (рисунок 6):

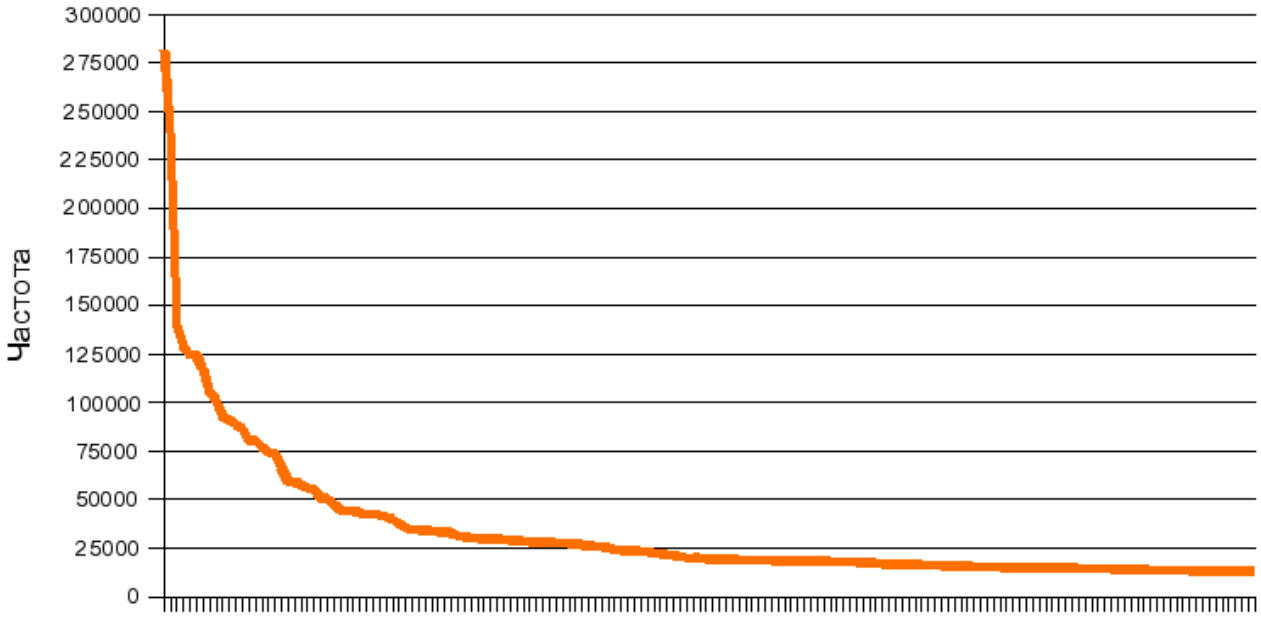

Рисунок 6

В этом распределении каждый i-ый элемент встречается в i раз меньше последующего, что можно выразить формулой:

$$f(i) = \frac{\lambda^i}{i!}$$

где параметр λ - коэффициент, выбираемый для каждого текста отдельно, в зависимости от его длины и разнообразия слов в нем, а i – номер элемента.

Эта формула имеет вид функции вероятности Пуассона:

$$P(x = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

Это также подтверждается экспериментальным путем, при помощи построения графиков распределения слов в тексте, при достаточно малых значениях λ, так как при увеличении λ, функция примет вид

$$p(k) \approx \frac{1}{\sqrt{2\pi\lambda}} exp\left(-\frac{(k-\lambda)^2}{2\lambda}\right)$$

Что является видом функции распределения Гаусса.

При помощи созданной мной утилиты, я построил несколько графиков распределения слов в небольшом наборе текстов (рисунок 7):

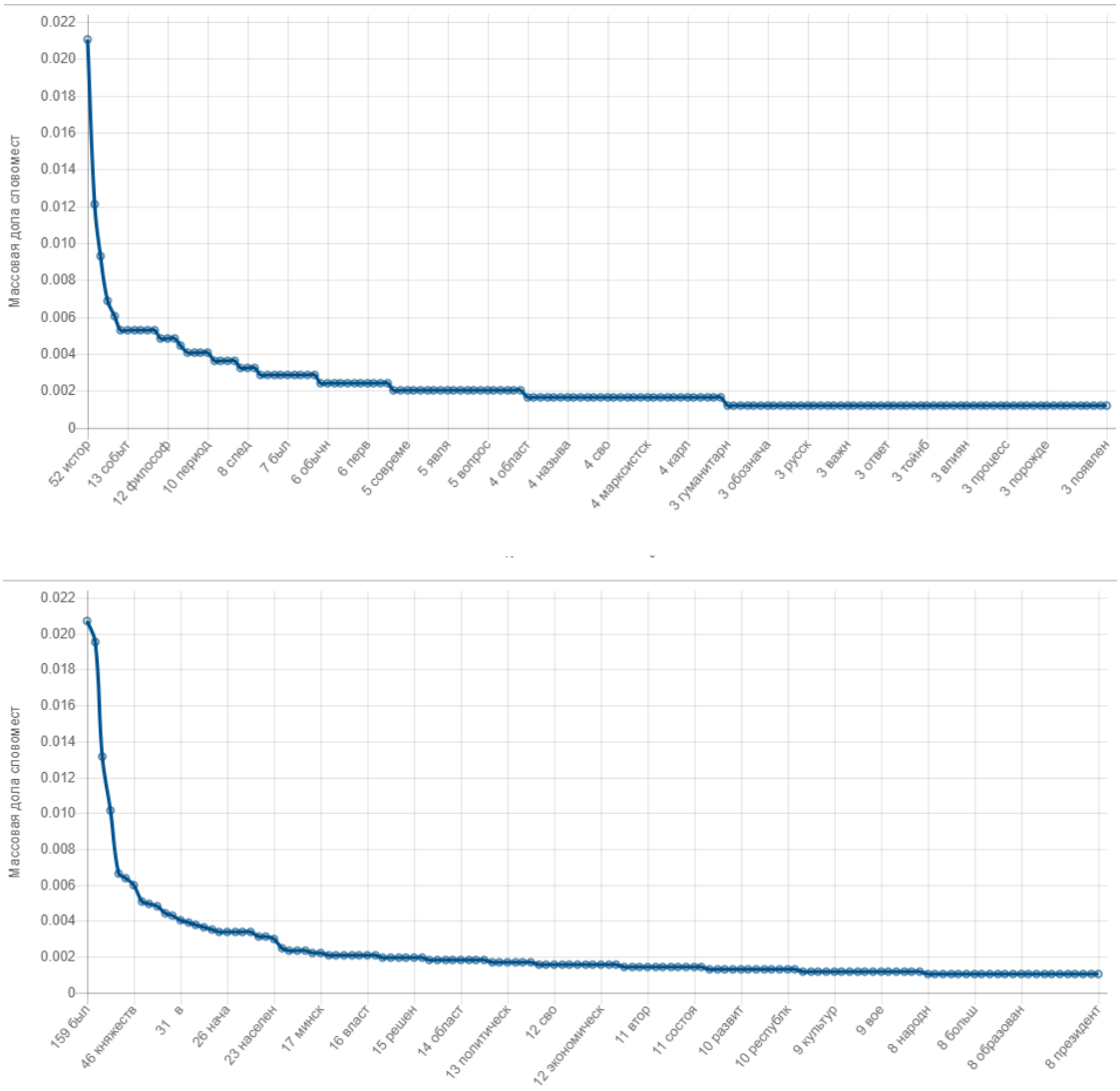

Рисунок 7

Для отрисовки этих графиков распределения я вычислил координаты точек, где на вертикальной оси находится массовая доля словомест на данное слово, а на горизонтальной – количество повторений данного слова в тексте.

После этого был построен кубический сплайн, который позволил интерполировать многочлен с корнями в найденных точках и по нему был построен график.

А именно:

Искомая функция задана на отрезке [a, b], разбитым найденными ранее значениями x_i на части и $a = x_0 < x_1 < \ldots < x_n = b$.

Кубическим сплайном дефекта 1 будет называться функция *S(x)*, которая:

- На каждом отрезке $[x_{i-1}, x_i]$ является многочленом степени не выше 3.

- Имеет непрерывные первую и вторую производные на всем отрезке [a, b].
- В точках x_iвыполняется равенство $S(x_i) = f(x_i)$.

На каждом отрезке $[x_{i-1}, x_i]$ является многочленом степени не выше 3, запишем его в виде:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Тогда

$$S_i(x_i) = a, S_i'(x_i) = b_i, S_i''(x_i) = 2c_i$$

Условия непрерывности всех производных до второго порядка включительно записываются в виде:

$$S_i(x_{i-1}) = S_{i-1}(x_{i-1})$$
$$S_i'(x_{i-1}) = S_{i-1}'(x_{i-1})$$
$$S_i''(x_{i-1}) = S_{i-1}''(x_{i-1})$$

А условия интерполяции в виде

$$S_i(x_i) = f(x_i)$$

Обозначим в виде

$$h_i = x_i - x_{i-1}, f_i = f(x_i)$$

Отсюда получим формулы для вычисления коэффициентов естественного сплайна:

$$a_i = f(x_i)$$
$$d_i = \frac{c_i - c_{i-1}}{3h_i}$$
$$b_i = \frac{a_i - a_{i-1}}{h_i} + \frac{2c_i + c_{i-1}}{3} h_i$$
$$c_{i-1}h_i + 2c_i(h_i + h_{i+1}) + c_{i+1}h_{i+1} = 3\left(\frac{a_{i+1} - a_i}{h_{i+1}} - \frac{a_i - a_{i-1}}{h_i}\right)$$

Если учесть, что $c_0 = c_N = 0$, то вычисление коэффициентов можно провести при помощи метода прогонки для трехдиагональной матрицы (матрицы Якоби).

Таким образом можно получить функцию, по которой можно нарисовать графики, показанные на рисунке 7.

Из этих графиков, как было сказано выше, можно выяснить, что слова в обозреваемых текстах распределяются по Пуассону.

Определим используемые далее обозначения:

- w – слово в текстовой коллекции;
- d – документ в текстовой коллекции;
- n_{dw}– частотность слова w в документе d;
- D – текстовая коллекция;
- T – множество выделяемых тем в текстовой коллекции D;
- W – множество уникальных слов в текстовой коллекции D (словарь);
- $\Phi = \{\varphi_{wt}\} = \{P(w|t)\}$ – матриц
- а скрытых распределений $P(w|t)$;
- $\Theta = \{\theta_{td}\} = \{P(t|d)\}$ – матрица скрытых распределений $P(t|d)$.

Рассмотрим каждый текстовый документ d как смесь неизвестных тем t, а каждую тему t – как распределение над словами.

Тогда распределение слов в документах происходит по правилу:

$$P(w|d) = \sum_{t} P(w|t)P(t|d),$$

где $P(w|t)$ и $P(t|d)$ – скрытые распределения слов по темам и тем по документам, а $P(w|d)$ – наблюдаемое распределение слов по документам.

Задача построения тематической модели заключается в восстановлении скрытых распределений $P(w|t)$ и $P(t|d)$ по известной коллекции документов D.

Метод Вероятностного Латентного Семантического Анализа решает поставленную задачу методом максимума правдоподобия:

$$log \prod_{d \in D} \prod_{w \in d} P(w|d)^{n_{dw}} = \sum_{d \in D} \sum_{w \in d} n_{dw} log \sum_{t \in T} \varphi_{wt} \theta_{td} \rightarrow \max_{\Phi, \Theta}$$

При ограничениях неотрицательности и нормировки:

$$\varphi_{wt} = P(w|t) \geq 0, \sum_{w \in W} \varphi_{wt} = 1, \theta_{td} = P(t|d) \geq 0, \sum_{d \in D} \theta_{td} = 1$$

Поскольку тематическая модель зависит от нескольких скрытых переменных, для нахождения оценок максимального правдоподобия параметров Φ и Θ используется EM-алгоритм (Expectation-Maximization). Это итеративный алгоритм, каждая итерация которого состоит из двух шагов, повторяющихся до сходимости или до заданного числа итераций:

1. E-шаг. На данном шаге вычисляется ожидание значение функции правдоподобия, при этом скрытые переменные рассматриваются как наблюдаемые. В рассматриваемой задаче условные вероятности

P(t|d,w) для всех тем *t*, документов *d* и слов *w* вычисляются через скрытые параметры φ_{wt} и θ_{td} по формуле Байеса:

$$P(t|d,w) = \frac{P(w,t|d)}{P(w|d)} = \frac{P(w|t)P(t|d)}{P(w|d)} = \frac{\varphi_{wt}\theta_{td}}{\sum_{s\in T}\varphi_{ws}\theta_{sd}}$$

2. М-шаг. На данном шаге находится оценка максимального правдоподобия, тем самым увеличивается ожидаемое правдоподобие. В рассматриваемой задаче частотные оценки условных вероятностей вычисляются путём суммирования счётчика $n_{dwt} = n_{dw}P(t|d,w)$:

$$\varphi_{wt} = \frac{\sum_{d\in D} n_{dwt}}{\sum_{d\in D}\sum_{w\in d} n_{dwt}}, \theta_{td} = \frac{\sum_{w\in D} n_{dwt}}{\sum_{w\in W}\sum_{t\in T} n_{dwt}}$$

Таким образом мы получили матрицы φ_{wt} и θ_{td} искомых распределений *P(w|t)* и *P(t|d)* соответственно.

Теперь можно оценить качество полученной тематической модели при помощи одного из самых известных критериев – Перплексии.

Перплексия представляет собой меру несоответствия модели *P(w|d)* словам *w*, наблюдаемым в документах коллекции, и определяется через логарифм правдоподобия:

$$Perplexity(D) = e^{(-\frac{1}{n}\sum_{d\in D}\sum_{w\in d} n_{dw}\log P(w|d))}$$

где n – число всех рассматриваемых слов в текстовой коллекции, *D* – множество всех документов в коллекции, *ndw* – частотность слова *w* в документе *d*, *P(w|d)* – вероятность появления слова w в документе *d*.

Чем меньше значение перплексии, тем лучше модель предсказывает появление слов *w* одной темы в документах коллекции *D*.

Другим общепринятым методом оценки качества построенной модели является экспертные оценки. Экспертам предоставляются списки слов и словосочетаний, упорядоченных по мере принадлежности к неопределенной теме. После этого эксперты классифицируют эти списки по темам, и чем меньше неклассифицированных списков, тем модель точнее.

Пример таких списков:

Список	Тема
Беларусь, БССР, Польша, княжество, ВКЛ, население	История Беларуси
Быть, человек, люди, год, когда, время, женщина	---

Таким образом, работа приложения будет выглядеть как на рисунке 8:

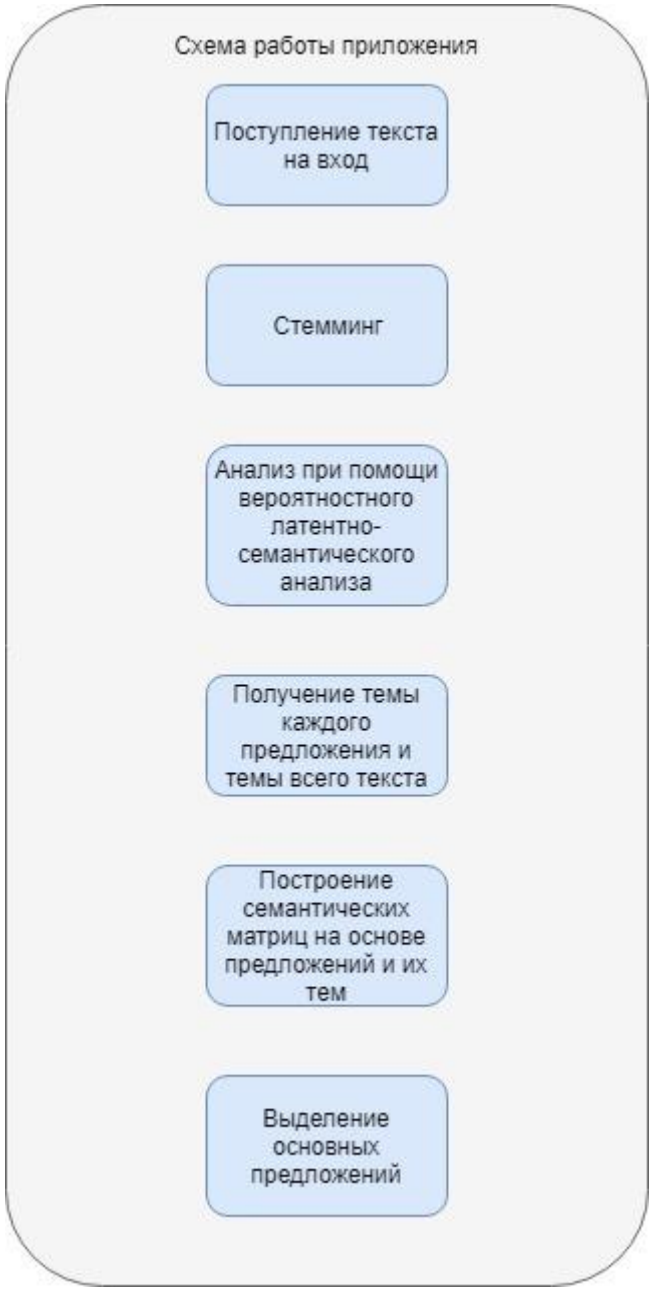

Рисунок 8

ГЛАВА 4 РАЗРАБОТКА ПРИЛОЖЕНИЯ

Разрабатываемое приложение будет строиться на основе микросервисной архитектуры. Микросервисная архитектура основана на взаимодействии маленьких, независимых и легко изменяемых модулей, которые называются микросервисы.

По сравнению со стандартной, так называемой монолитной архитектурой приложения микросервисная архитектура имеет как ряд преимуществ:

- Любой из модулей легко заменить или изменить, так как все микросервисы слабо связаны между собой
- Каждый микросервис имеет свою область ответственности, а следовательно выполнять одну функцию, что поможет тестировать приложение
- Каждый микросервис может быть реализован при помощи различных языков программирования, выполняться в различных средах виртуализации и управляться при помощи различных операционных систем.
- При высокой нагрузке на систему более просто горизонтально масштабировать микросервисное приложение, нежели монолитное.

Однако, существует и ряд недостатков, к примеру:

- Необходимо определить формат сообщений и стабильное соединение между различными микросервисами для обеспечения стабильной работы системы, чего в монолитных системах делать не требуется.
- Для эффективного использования ресурсов необходимо балансировать нагрузку и отказоустойчивость с правильным исполнением масштабируемости системы.
- Более трудоёмкая разработка приложения.

Разрабатываемое приложение будет состоять из следующих типов микросервисов:

- Авторизационный и аутентификационный микросервис
- Микросервис для отдачи статики на фронтальную часть приложения
- Микросервис для распределения нагрузки на другие микросервисы, так называемый load balancer
- Микросервис для обработки звуковых файлов
- Микросервис для обработки текстовых файлов

Существует большое количество решений для разработки микросервисных приложений, в этой работе будет использоваться платформа Azure Service Fabric.

Azure Service Fabric будет использоваться так как:

- Эта платформа имеет большое количество встроенных решений помогающих в сообщении различных типов микросервисов.
- Эта платформа предоставляет встроенный микросервис для распределения нагрузки на другие микросервисы.
- Автор приложения уже знаком с этой платформой и имеет опыт разработки приложений на ней.

Схематично конечное приложение можно представить как на рисунке 9 (Без БД).

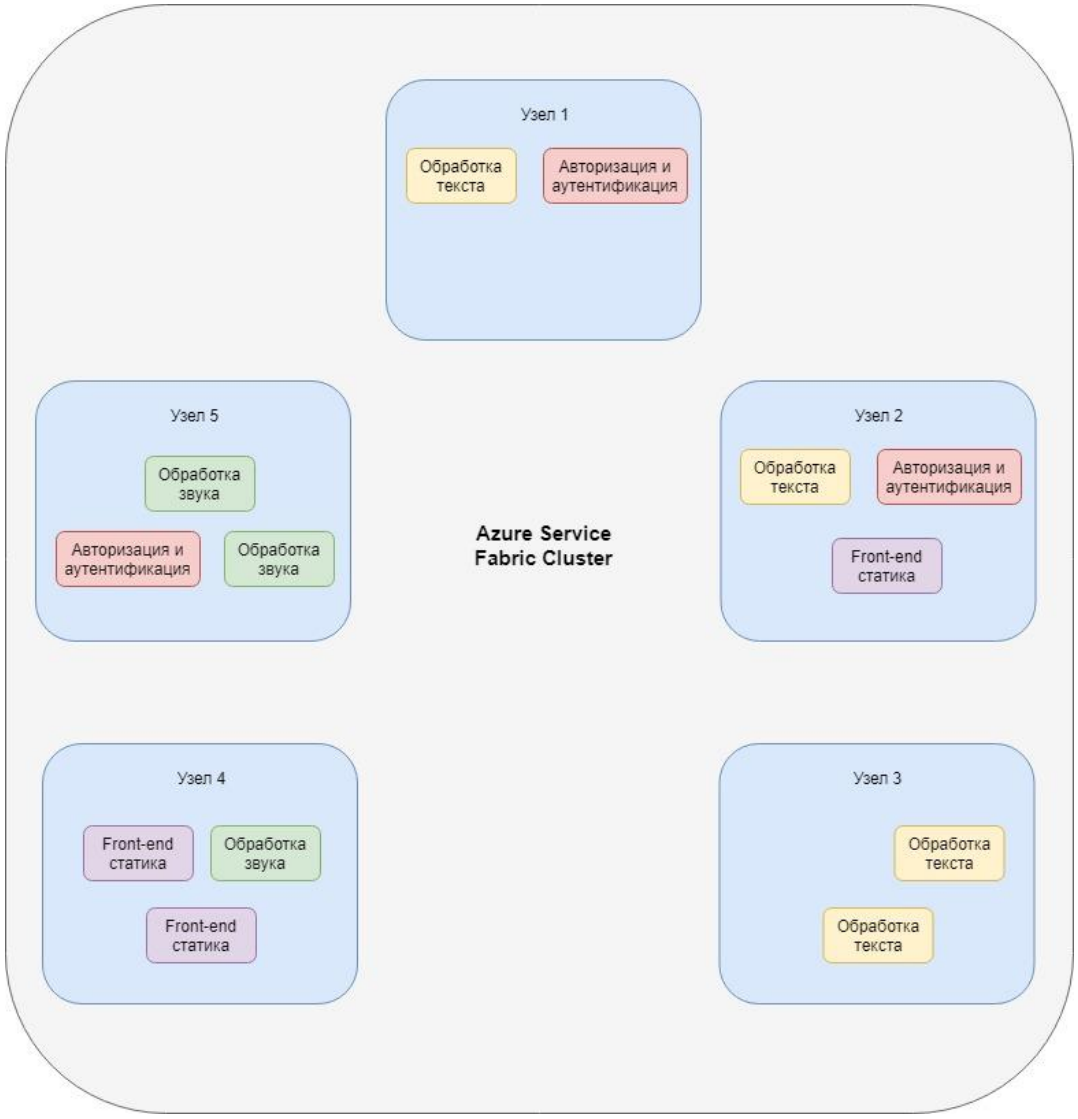

Рисунок 9

Далее, каждый из микросервисов можно также схематично представить при помощи диаграммы, например микросервис для распознавания речи иметь вид как на рисунке 10

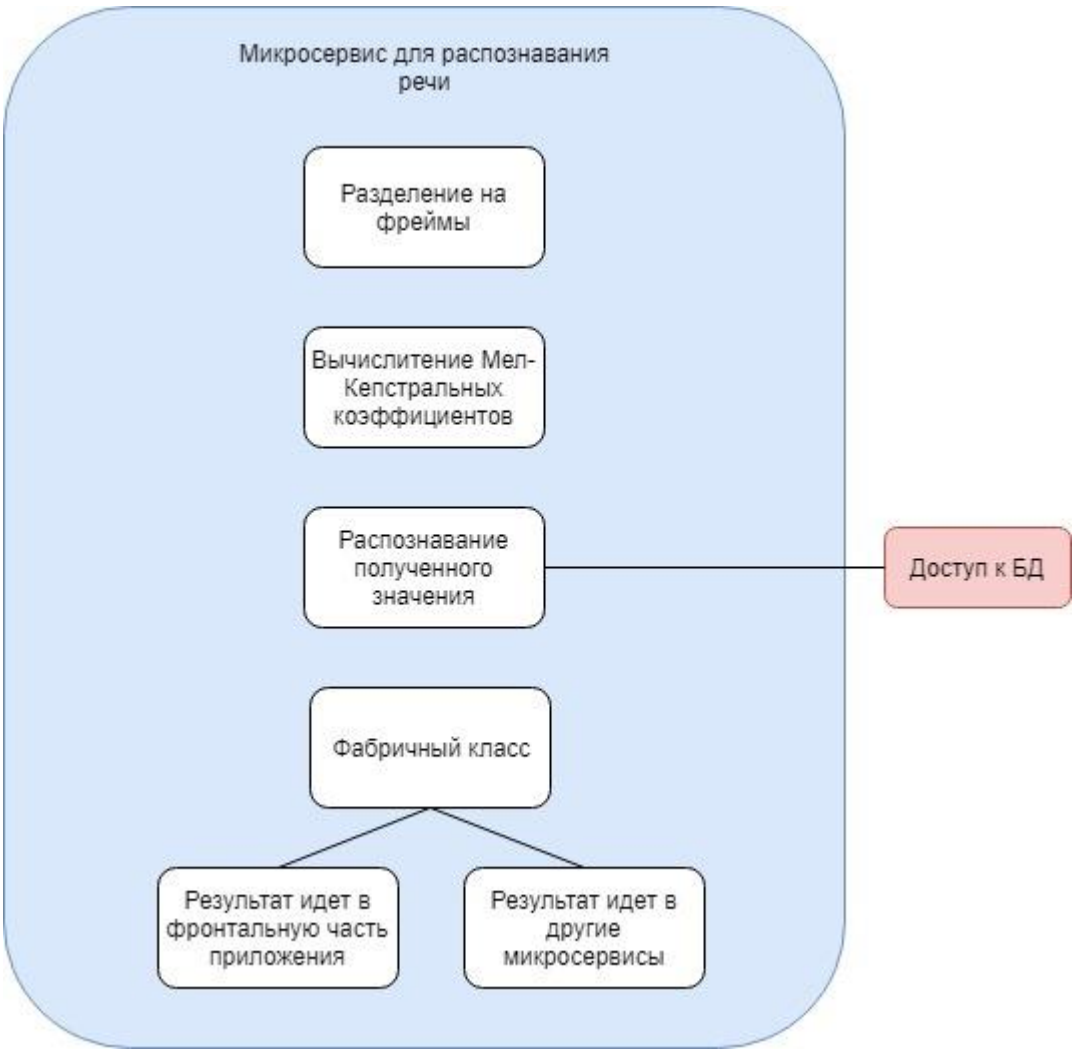

Рисунок 10

Авторизационный и аутентификационный микросервис можно представить при помощи схемы изображенной на рисунке 12:

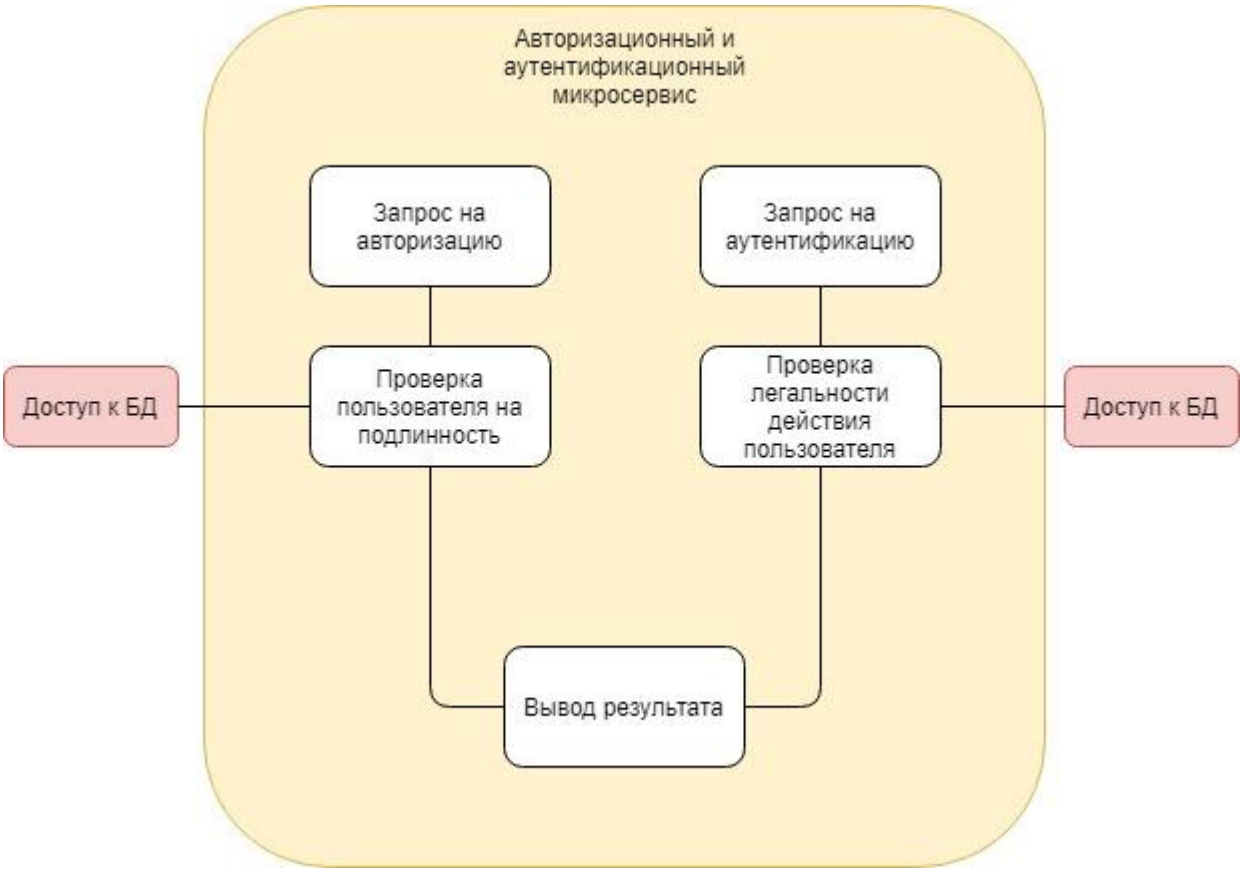

Рисунок 12

Микросервис для анализа текста, в свою очередь будет иметь вид как на рисунке 12:

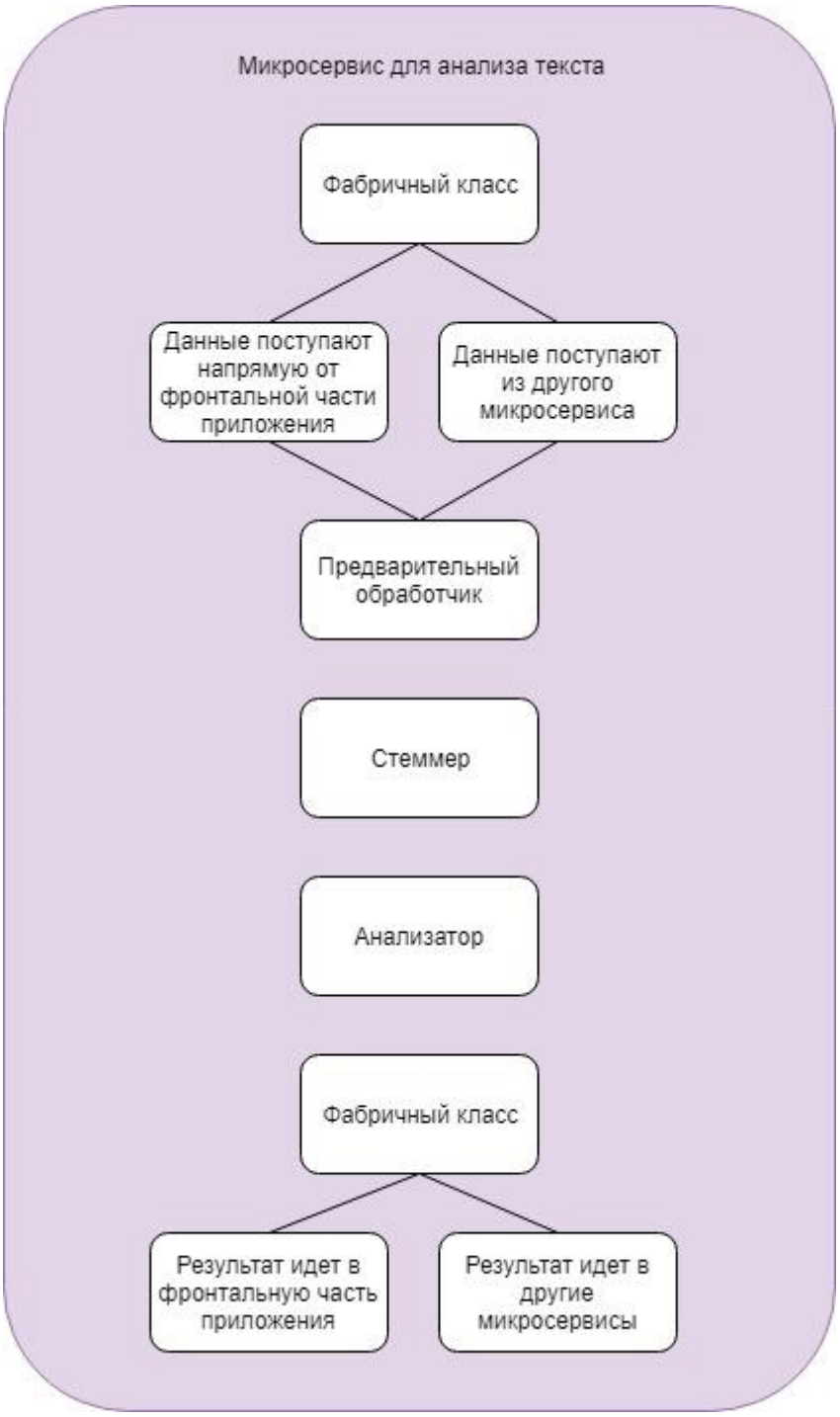

Рисунок 12

Для разработки приложения использовался язык программирования C# (.NET Core) , так как:

- Это объектно-ориентированный язык программирования. Это означает, что можно достаточно просто описывать абстрактные конструкции на основе предметной области и реализовывать взаимодействие между ними.
- Он работает на базе платформы .NET, что означает, что это приложение можно будет запустить на любой операционной системе, на которой

установлена данная платформа. За счет этого также возможно увеличение производительности программы, так как будут использованы команды специфичные для каждого процессора.
- Существует большое количество библиотек и шаблонов проектирования, которые упрощают разработку.
- Существует большое количество хороших инструментов разработки, таких как Visual Studio, которые предоставляют большое количество различного рода возможности.
- Автор данной дипломной работы знаком с этим языком больше всего.

В качестве основной СУБД использовалась MSSQL. Основным языком запросом в этой системе является Transact-SQL. Это реляционная СУБД, которая характирезуется такими особенностями как:

- Надежность и безопасность данных – в MSSQL реализована возможность шифрования данных.
- Производительность.
- Простота – с MSSQL относительно легко работать и вести администрирование.

Реляционная модель предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая из строк хранит в себе отдельный объект, а в столбцах хранятся его атрибуты.

Для идентификации каждой строки в рамках таблицы применяется первичный ключ. В качестве такого ключа могут выступать один или несколько столбцов. При помощи такого ключа можно однозначно определить строку, к которой этот ключ относится.

При помощи этих ключей строки разных таблиц могут быть связаны, то есть между таблицами могут быть организованы связи.

В процессе разработки автор использовал большое количество различных библиотек, например:

- Autofac – контейнер инверсии управления. Инверсия управления – важный принцип программирования, который используется для создания независимых структур. Такое архитектурное решение позволяет улучшить расширяемость системы, а также позволяет улучшить тестируемость системы. Autofac – легко используемы контейнер с поддержкой Azure Service Fabric.
- Automapper – библиотека, позволяющая при помощи заранее заданных конфигураций проецировать модели, содержащие данные, друг на друга. Это полезно при использовании патерна «Репозиторий», так как позволяет отделять логику и модели отправки данных от логику и модели обработки данных.

- Swashbuckle.Swagger – библиотека, созданная для упрощения разработки и позволяющая визуализировать все возможные запросы в работающее приложение.
- Fluentvalidation – библиотека, позволяющая валидировать входные данные при помощи цепочечного вызова интерфейсов.
- Jquery – Библиотека, продоставляющий удобный API для работы с AJAX, которая фокусируется на взаимодействии JS и HTML.

Было использовано большое количество структурных и архитектурных шаблонов, например так называемый Generic Repository.

При этом часть структуры приложения, отвечающей за работу с базой данных и обработкой данных будет выглядеть как на рисунке 13:

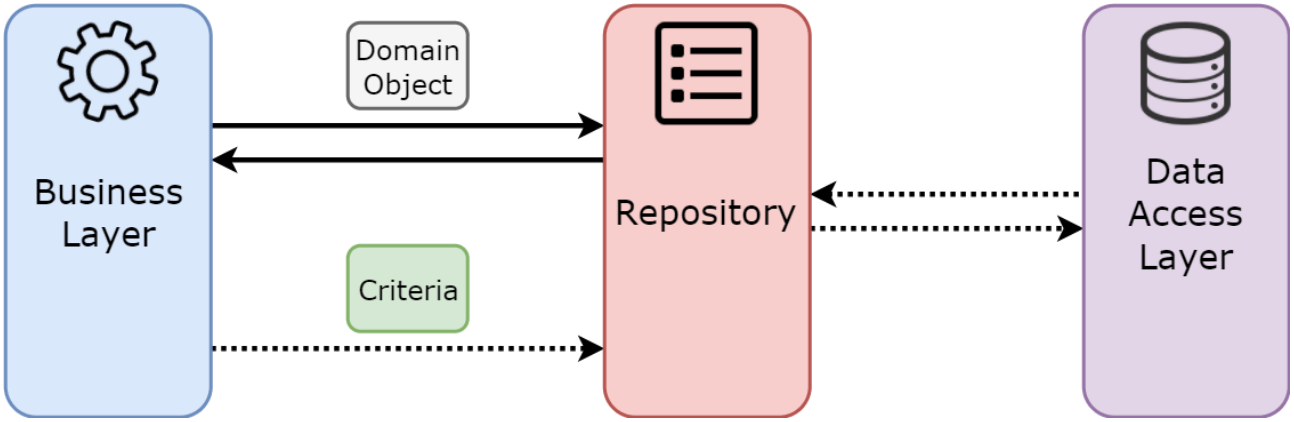

Рисунок 13

Где Business Layer – часть приложения, отвечающая за обработку данных, Data Access Layer – СУБД, в которой хранятся данные, а Repository(так называемое «Хранилище») – абстракция, которая позволяет абстрагироваться от конкретной СУБД и модели данных, для того, чтобы при надобности была возможность с меньшими усилиями заменить их на другую СУБД и модель использования данных.

Также был использован архитектурных шаблон SOLID, который состоит из 5 частей:

- Принцип единственной ответственности (The Single Responsibility Principle). Каждый объект приложения должен иметь только одну ответственность и эта ответственность должна быть полностью инкапсулирована в класс. Этот принцип предлагает разделять универсальный классы на конкретные, что сделает их более простыми и легкими в обслуживании и написании. Таким образом система становится легко изменяемой, расширяемой и тестируемой.
- Принцип открытости/закрытости (The Open-Closed Principle). «Программные сущности должны быть открыты для расширения, но закрыты для модификации». Это значит, что спецификации интерфейсов

могут быть переиспользованы путем их наследования, но существующие реализации изменяться не должны. Это особенно важно в производственной среде, так как модификация существующего кода требует дополнительного просмотра кода, тестирования и прочих процедур, в то время как код, который подчиняется данному принципу не изменяется при расширении системы и поэтому не требует таких трудозатрат. Также нарушение этого принципа влечет за собой нарушение принципа единственной ответственности.

- Принцип подстановки Барбары Лисков (The Liskov Substitiution Principle). «Функции которые используют базовый тип, должны иметь возможность использовать все подтипы базового типа, не зная об этом». Это означает, что если у типа T существует подтип S, то вместо каждого использования типа T возможно использовать тип S без каких-либо изменений в программе. Это возможно только в том случае, если наследующие классы не изменяют, а дополняют базовые. Этот принцип является одним из наиболее важных критериев для оценки качества принимаемых решений при построении иерархий наследования.
- Прицнип разделения интерфейса (The Interface Segregation Principle). «Много интерфейсов, специально предназначенных для одной задачи лучше одного универсального». Если этот принцип соблюден, то это дает системе возможность оставаться гибкой даже при внесении изменений в логику ее работы. Также нарушение этого принципа влечет за собой нарушение принципа единственной ответственности.
- Принцип инверсии зависимостей (The Dependency Inversion Principle). «Абстракции не должны зависеть от деталей. Детали должны зависеть от абстракций. Модули верхних уровней не должны зависеть от модулей нижних уровней, обы типа модулей должны зависеть от абстракций». Это означает, что классы приложения не должны содержать зависимостей на реализации, а на абстракции, что позволяет легко подменять реализации одну на другую. Это облегчает тестирование приложения.

ГЛАВА 5. ОЦЕНКА ПОЛУЧЕННЫХ ЗНАЧЕНИЙ

В результате некоторых тестов, например показанном на рисунке 14:

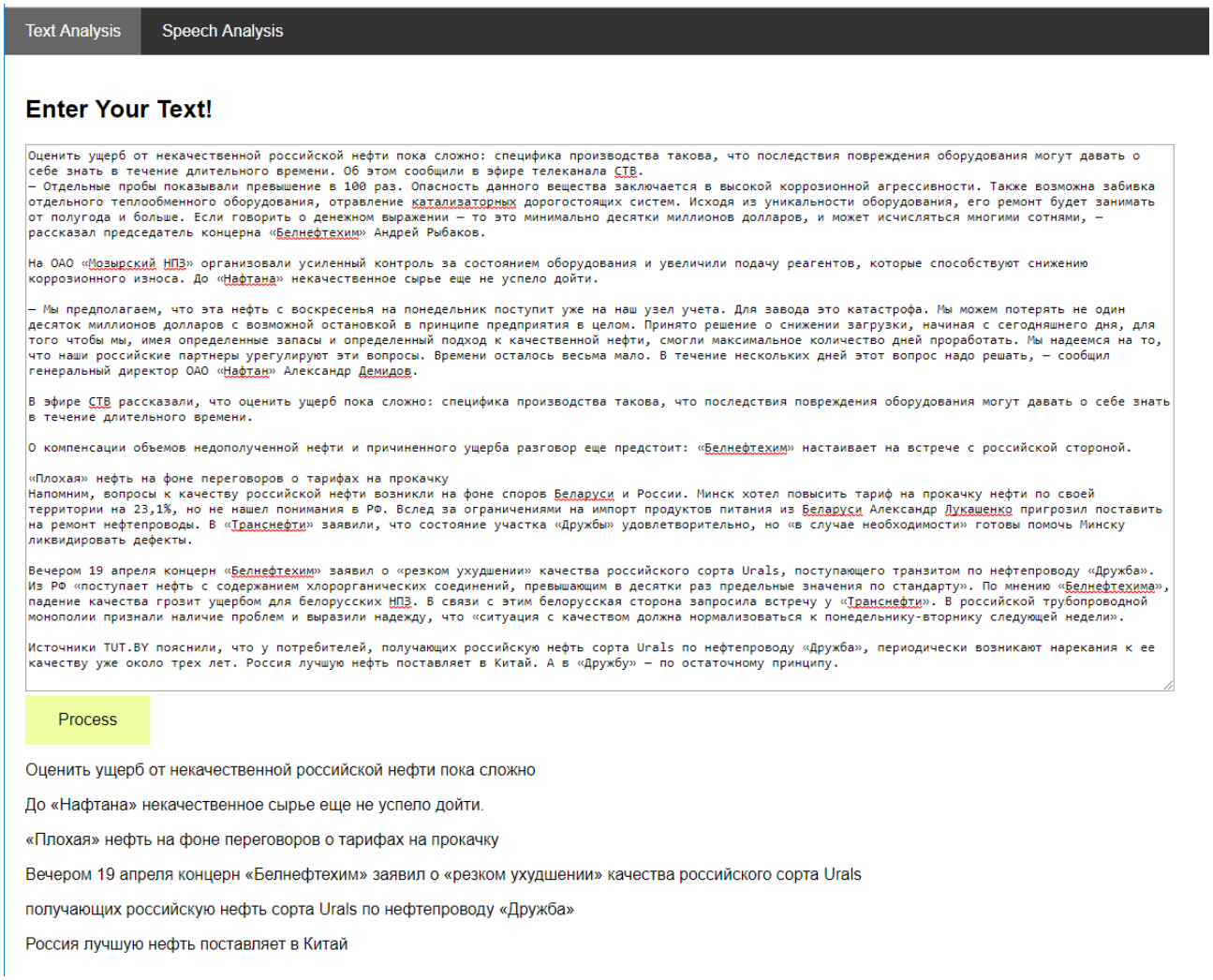

Рисунок 14

Было выявлено, что работа используемого алгоритма недостаточно точна для конспектирования текста, но достаточно точна для классификации текста и его частей и составлении матриц сингулярного разложения. Классификация – одна из важнейших задач анализа информации, и такие «промежуточные» значения могут очень помочь в анализе информации далее. С нахождением более удачных коэффициента λ и матрицы разложения V^t для каждого текста в отдельности точность этого алгоритма возрастает. Для нахождения коэффициента λ и матрицы разложения V^t можно использовать обученную нейронную сеть, сгенерированную при помощи генетических алгоритмов. Тогда задача классификации будет решена, а также будут построены матрицы сингулярного разложения и результат работы этого алгоритма можно будет

использовать в других. Таким образом уже успешно работает Azure Text Analytics API и возможно другие существующие сервисы.

ЗАКЛЮЧЕНИЕ

В ходе данного дипломного проекта:

а) Были изучены методы и алгоритмы обработки естественного языка

б) Был проведен анализ современных алгоритмов для решения проблемы компьютерного анализа естественных языков.

В данном дипломном проекте были рассмотрены некоторые алгоритмы обработки естественного языка и распознавания речи. В приложениях к ней реализованы некоторые из этих алгоритмов и рассмотрена их эффективность и использование на практике.

Также были поставлены цели на будущее по развитию данных приложений и изучению этой темы в дальнейшем, а именно:

1. Разработка и изучение более эффективных алгоритмов обработки естественного языка.
2. Реализация алгоритмов обработки текста.
3. Реализация алгоритмов распознавания речи.
4. Реализация более удобного пользовательского интерфейса.

При анализе существующих программ, использующих реализованный алгоритм было выявлено, что данный алгоритм показывает высокие результаты, так как он дает возможность достоверно классифицировать и составлять матрицы сингулярного разложения для текстов большого размера и больших коллекций документов, однако использование только этого алгоритма не обладает настолько большой эффективностью, как использование нескольких алгоритмов анализа текста сразу вместе с подбором коэффициентов для алгоритмов для каждого выбранного текста отдельно при помощи специально обученных под конкретные темы нейронных сетей.

СПИСОК ИСПОЛЬЗОВАННЫХ МАТЕРИАЛОВ

1. Латентно-семантический анализ[Электронный ресурс]. - https://habrahabr.ru/post/110078/

2. Распознавание речи [Электронный ресурс]. - https://habrahabr.ru/post/226143/

3. speech-recognizer[Электронный ресурс]. - https://github.com/krestjaninoff/speech-recognizer/tree/article-basic

4. Russian stemming algorithm[Электронный ресурс]. - http://snowball.tartarus.org/algorithms/russian/stemmer.html

5. Стеммер Портера для русского языка[Электронный ресурс]. - http://www.algorithmist.ru/2010/12/porter-stemmer-russian.html

6. Window function[Электронный ресурс]. - https://en.wikipedia.org/wiki/Window_function#Triangular_window

7. АНАЛИЗ МЕТОДА МЕЛ-ЧАСТОТНЫХ КЕПСТРАЛЬНЫХ КОЭФФИЦИЕНТОВ ПРИМЕНИТЕЛЬНО К ПРОЦЕДУРЕ ГОЛОСОВОЙ АУТЕНТИФИКАЦИИ[Электронный ресурс]. - http://publikacia.net/archive/2015/10/1/24

8. Распознавание речи[Электронный ресурс]. - https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1 %80%D0%B5%D1%87%D0%B8

9. CLR via C#[Учебное пособие по программированию] – Джеффри Рихтер

10. Вероятностный латентно-семантический анализ [Электронный ресурс]- http://www.machinelearning.ru/wiki/index.php?title=%D0%92%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%D1%81%D1%82%D0%BD%D1%8B%D0%B9_%D0%BB%D0%B0%D1%82%D0%B5%D0%BD%D1%82%D0%BD%D1%8B%D0%B9_%D1%81%D0%B5%D0%BC%D0%B0%D0%BD%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BD%D0%B0%D0%BB%D0%B8%D0%B7