**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

# РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

## *К КУРСОВОМУ ПРОЕКТУ*

## *НА ТЕМУ:*

## *«Исследование методов обработки и анализа текстовых данных и графах знаний»*

Студент ИУ5-33М (Группа) _____ (Подпись, дата) Д.Р. Громоздов (И.О.Фамилия)

Руководитель _____ (Подпись, дата) Ю.Е. Гапанюк (И.О.Фамилия)

Консультант _____ (Подпись, дата) _____ (И.О.Фамилия)

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение**
**высшего образования**
**«Московский государственный технический университет имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой _____
(Индекс)

_____ _____
(И.О.Фамилия)

« _____ » _____ 20 ____ г.

# З А Д А Н И Е
## на выполнение курсового проекта

по теме « Исследование методов обработки и анализа текстовых данных и графах знаний» __
_____

Студент группы __ИУ5-33М__

_____Громоздов Данила Романович_____
(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

__исследовательская_____

Источник тематики (кафедра, предприятие, НИР) _____НИР_____

График выполнения НИР: 25% к _4_ нед., 50% к _8_ нед., 75% к 12 нед., 100% к 16_ нед.

Техническое задание: изучить литературу, содержащую информацию по теме НИР.____
Составить на её основе краткий обзор методов, применяемых при решении задач_____
анализа текстовых данных и графов знаний. Провести эксперименты по теме НИР._____

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на __ листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

_____
_____

Дата выдачи задания « _01_ » __сентября_____ 2022_ г.

**Руководитель НИР** _____ __Ю.Е. Гапанюк_
(Подпись, дата) (И.О.Фамилия)

**Студент** _____ __Д.Р. Громоздов
(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

# Оглавление

## Методы работы с графами знаний

Графы знаний – это база знаний, использующая топологию графа для отображения взаимосвязей между сущностями и явлениями. При этом связанные элементы представляют собой вершины графа, а поименованные рёбра отвечают за описание семантики этой взаимосвязи. В таких графах в качестве вершин могут выступать данные различных модальностей (т. е. разных типов представления информации, или способов получения данных от сенсоров: текст, изображения (в частности, цвет, яркость, глубина), дата-время, координаты и другие). Мультимодальное обучение – разновидность машинного обучения, в которой в качестве входных данных могут использоваться данные, представленные разными модальностями. При хорошем построении архитектуры нейронной сети использование графов знаний может позволить выявить больше характерных признаков, а следовательно, возможно улучшить её работу и возможности. Возможна обработка не только конкретного объекта, но и контекста, в котором он находится.

Для обучения на графовых представлениях используется вложение (отображение, embedding) его элементов (вершин и рёбер) в векторы или матрицы. Такой подход называется Вложением графа знаний (Knowledge graph embedding / KGE). На имеющихся данных изучаются возможные связи между сущностями, находящимися в вершинах этого графа, что позволяет достраивать неполные графы знаний, а также производить классификацию объектов, основываясь на доступных в графе признаках объектов и их взаимосвязях. Мультимодальное вложение графов знаний позволяет производить извлечение связей из графа, основываясь на информации, представленной несколькими модальностями одновременно. Данный подход, однако, требует приведения признаков всех модальностей (а для некоторых моделей и связей между ними) к общему пространству представлений для графа.

Графы знаний позволяют не просто использовать для анализа несколько модальностей, но также и отражать семантические взаимосвязи между ними.

Если обычные модели мультимодального обучения получают на вход информацию разных типов раздельно, то в случае использования графового представления данных информация будет передаваться в виде векторов, содержащих семантику взаимосвязи между объектами различной модальности.

Представление используемой для обучения информации в виде графа знаний можно рассматривать как один из этапов предобработки данных. Выявляются семантические зависимости между элементами, принадлежащими разным типам представления информации, таким образом эффективно расширяется признаковое пространство и упорядочиваются данные.

### *Общая схема эмбеддинга мультимодального графа знаний*

В данном разделе будет рассмотрена общая схема обработки мультимодальных графов знаний в ходе процесса обучения (рис. 1).

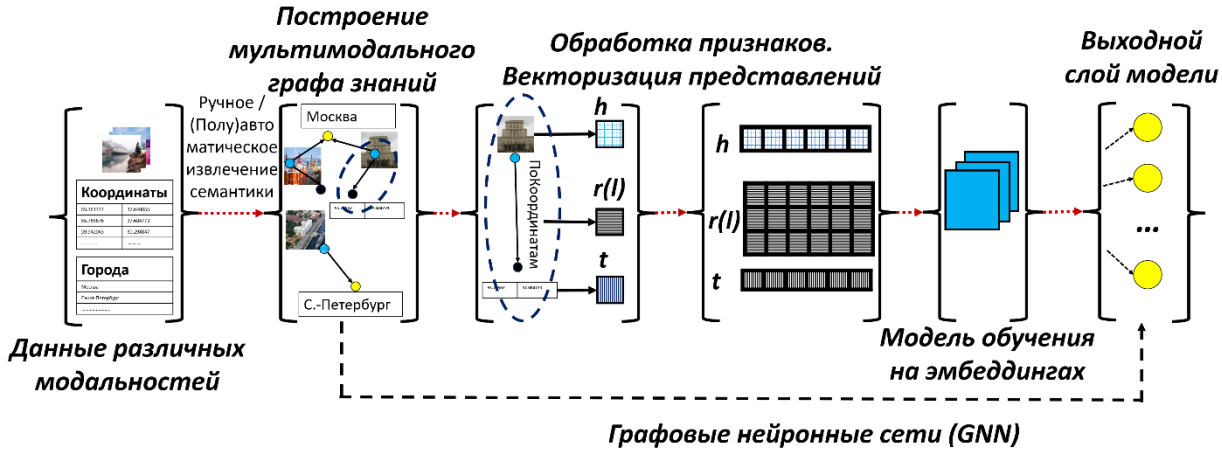



Рис. 1. Общая схема работы с мультимодальными графами знаний.

В начале берётся набор данных (датасет), содержащий информацию различных типов. На основе него извлекаются семантические связи между входящими элементами. Извлечение может осуществляться как вручную – связи полностью определяются человеком на основе знаний о предметной области, так и автоматически или полуавтоматически с использованием алгоритмов анализа и правил. Сущности и семантические связи между ними формируют граф знаний. При этом, по большей части рёбра в таких графах

имеют направленный характер из-за особенностей работы с семантическими связями (как видно на примерах *«МГТУ им. Баумана – НаходитсяВ – Москва»* и *«Москва – НаходитсяВ – МГТУ им. Баумана»*, они обозначают не одно и то же!).

Далее связи внутри графа знаний представляются в виде триплетов вида (*h, r, t*), встречается так же вариант (*h, l, t*), где *h* и *t* – вершины графа, а *r* (или *l*) – семантическая связь между ними. При этом учитывается направленность связи поскольку *h* – это «головная» (head) сущность, а *t* – «хвостовая» (tail) сущность. Субъектом связи является вершина *h*, а её объектом, соответственно, вершина графа *t*. Возвращаясь к предыдущему примеру, *«МГТУ им. Баумана»* было бы вложено в представление *h*, *«НаходитсяВ»* вкладывалось бы в связь *r*, *«Москва»*, объект связи, вложилась бы в представление *h*.

На данном этапе как раз критична разная модальность данных в графе, поскольку все вершины для работы модели должны находиться в одном пространстве представлений (в некоторых моделях также и связи) . Соответственно, для них применяются различного рода методы предобработки данных. Для обработки изображений может быть предварительно использована, например, свёрточная нейронная сеть , извлекающая признаки из изображения, которые затем могут быть вложены в пространство представлений сущности h или t.

Далее данные, представленные в виде (*h, r, t*) передаются на вход в модель обучения. При этом для каждого факта (*h, r, t*) вычисляется его оценочная (score) функция. Она представляет собой оценку вероятности того, что данный факт имеет место . Процесс обучение же заключается в максимизации общего значения этой функции (для верных фактов значение выше, для ложных утверждений ниже). Для перевода взаимодействия сущностей в связи используются разные модели, в том числе линейные архитектуры, рекуррентные нейронные сети и другие.

Выход будет соответствовать решаемой задаче: соответствующий второй элемент графа для предсказания связи; верность или неверность утверждения о

том, что данный тип связи может быть установлен для взятых двух сущностей и отношение элемента к конкретной семантической категории для задач классификации; удаление или не удаление возможного дубликата узла графа.

***Оценочные функции***

Как говорилось выше, оценочная (score) функция используется в моделях вложения графов знаний для оценки вероятности существования связи между некоторыми двумя сущностями. Такие функции можно разделить на два типа: основанные на оценке расстояний и на оценке сходства.

Первые оценивают расстояние между вершинами, сущностями, в графе, и на основе него предоставляют вероятность существования факта. К таковым относится, например, широко используемая модель TransE, KG2E, ManifoldE и другие.

Вторые дают оценку вероятности правильности факта на основании семантического сходства. К моделям, использующим оценочные функции данного семейства относят HolE, DistMult.

**Исследование методов представления графов**

Для того, чтобы оценить результат работы с графовыми моделями, можно воспользоваться средствами визуализации графов. Через графическое представление можно исследовать характер связей внутри графа, наличие или отсутствие кластеров и хабов, их примерное количество. Также можно узнать, является ли итоговый граф связным и/или плоским. Наличие различных сгруппированных структур в графе может свидетельствовать о наличии в нём разделения по тематикам связей или вершин. Это может помочь в формировании, например, гипер- или метаграфовых представлений данных.

Возможности отображения графов были оценены на наборе данных $ATOMIC_{20}^{20}$, представляющем из себя базу знаний в виде условий если-то, отражающих логический вывод для ситуаций реального мира. Часто этот набор данных используется для задач commonsense reasoning.

Этот набор устроен как набор триплетов, характеризующийся объектом, субъектом и типом их логической связи. В качестве объектов и субъектов могут выступать люди, события или эмоциональное состояние. Такое устройство датасета делает его очень удобным для представления его в виде графа.

Для исследования визуализации графов была выбрана Python библиотека igraph. Результат работы функций этого модуля представлен на

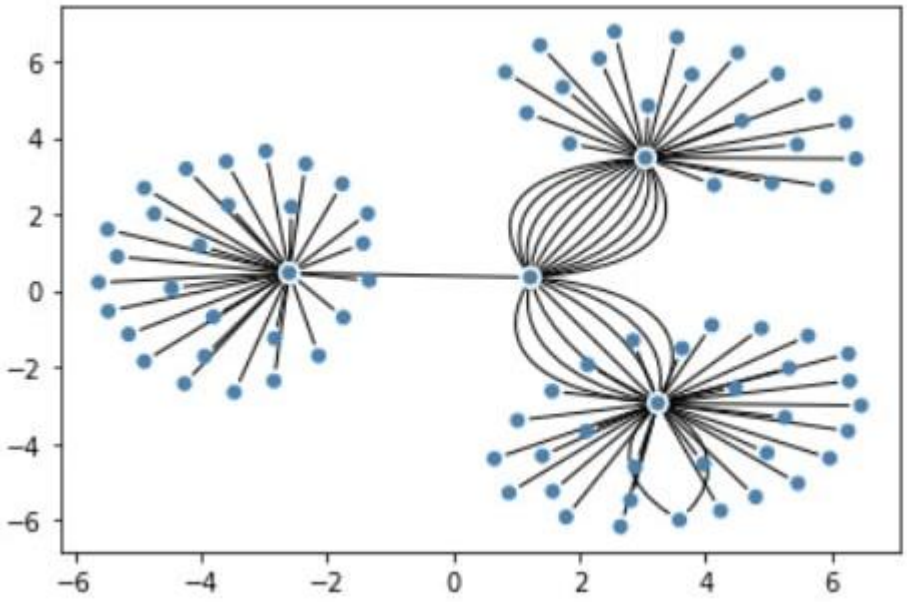



Рис. 2. Визуализация выборки данных из $ATOMIC_{20}^{20}$ с помощью *igraph*.

Уже на небольшой выборке данных мы видим, что в данной системе скорее всего можно будет наблюдать хабы, имеющие множество связей с другими вершинами. Также можно выделить следующую особенность: между двумя вершинами может существовать сразу несколько связей.

Эта библиотека предоставляет инструменты и для вычисления различных характеристик, например, матрицы смежности для вершин, которые могут быть использованы в алгоритмах работы с графовыми данными, в том числе и для машинного обучения.

Для более точного анализа следует провести визуализацию сразу всего набора данных, однако это процесс, затрачивающий много ресурсов. Следующим этапом может быть использование мощностей среды, например, Google Colab для обработки большего количества вершин и рёбер.

Дополнительная проблема, для которой важно разработать решение – это читаемость графов с большим количеством вершин и рёбер, особенно, если они содержат текстовые данные.

### Методы обработки текстовых данных

Один из способов работы с мультимодальными данными – это предварительно обработать их и привести к общему виду с другими данными. Для этого подходит алгоритм эмбеддинга. В литературе есть примеры, где предварительно данные приводят к векторному виду. Одни из передовых моделей в векторизации текстовых данных – это представители семейства моделей word2vec, получающие на вход массив токенов, извлечённых из текста, а на выходе дающий вектора для каждого токена в n-мерном пространстве.

Представим цепочки добавленной стоимости для процесса обучения исследуемой модели. Сначала происходит импорт необходимых библиотек обработки данных. Они содержат способы представления данных, функции и классы, описывающие работу алгоритмов машинного обучения. Далее в пространство, в котором будет происходить обучение происходит загрузка в переменную значений данных в читаемом для алгоритма МО представлении. На следующем этапе происходит токенизация. Это процесс преобразования с помощью алгоритма-токенизатора данные типа строка в отдельные *токены*, которые будут восприниматься последующими моделями как единица текста. В данном случае используется алгоритм WordPunctTokenizer, который разбивает строку на отдельные слова на основе табуляции и пунктуации.

Далее данные в таком виде поступают на модель векторизации, в качестве которой здесь используется модель word2vec. Данное семейство моделей представляет собой комбинацию двух типов архитектур нейронных сетей: CBoW (Continuous Bag of Words) и Skip-gram. Первая модель предсказывает следующий токен исходя из имеющейся последовательности токенов (прогнозирует следующее слово в предложении) на основе «мешка

слов» (Bag of Words), т.е. на основе всего набора токенов, встречающихся в наборе данных, и их частоты встречаемости, CBoW добавляет «непрерывность», то есть ещё учитывает контекст. Особенность word2vec в том, что она с помощью этих архитектур может формировать векторные представления слов. Каждый токен в процессе обучения присваивается некоторому вектору в n-мерном пространстве. На основе взаимного расположения этих векторов можно делать выводы о близости значений слов и вероятности их появления рядом в контексте.

Используемая в исследуемой ИС модель word2vec имеет в основе архитектуру CBoW, поэтому используется алгоритм, разбивающий на токены, соответствующие словам. Обучение модели проходит в течение 5 эпох (проходов по всему обучающему набору данных), она игнорирует слова, встречающиеся реже 10 раз, максимальное количество слов контекста = 10. Не обязательно использовать возможность предсказания токенов по контексту предшествующих ему, в контексте исследуемой ИС более важно получить от этой модели векторные представления. Это многомерное пространство векторов можно делить различными плоскостями для решения целевой задачи: классификации.

Следующей в цепи процессов располагается, собственно, модель классификации. В исследуемой системе это алгоритм KNN (k-Nearest Neighbors) или k-ближайших соседей. Эта модель относит выборочное значение к тому или иному классу на основании того, к какому классу принадлежат его ближайшие соседние значения (их количество и обозначает параметр $k$). На практике удовлетворительные результаты классификации получаются при $k \approx 3$. При $k$ больше 10 часто не происходит улучшения, но временные затраты на алгоритм классификации увеличиваются. Путём анализа работы алгоритма при разных значениях для исследуемой системы параметр $k$ приняли равным 5.


Рис. 3. ARIS-диаграмма цепочки добавленной стоимости для модели МО

Далее к системе хранения данных также посылается запрос на данные для считывания по аналогичному каналу передачи данных. Результат запроса от БД передаётся обратно по данному каналу. Результат запроса подаётся на обученную модель машинного обучения (в ассоциации с моделью векторизации), которая производит обработку полученных данных.

Важно отметить, что этап непосредственно обучения для модели машинного обучения  происходит в общем случае один раз, соответственно, во время первой итерации обработки данных.

Оценка времени машинного обучения:

$$t_{\text{МО}} = t_{\text{обучения}} + t_{\text{кода}} + t_{\text{счит.}} \quad (1)$$

***Теоретическая оценка значений времени***

Для начала оценим времена передачи данных по каналу связи. Вот формулы для их вычисления:

$$t_{\text{канала}_1} = \frac{(q + m_{sup}) \cdot \lfloor \log_2 N \rfloor + i_{id}}{V}$$

$$t_{\text{канала}_2} = \frac{k \times \sum_a [Avg(m_a) \cdot \lfloor \log_2 N_a \rfloor] + i_{id}}{V}$$

В формуле для времени передачи текста запроса нам неизвестны $q, m_{sup}, N, V$ и $i_{id}$.

Значение параметра $q$ определяем исходя из общего вида простого запроса на чтение:

```
SELECT {атрибуты} FROM {название таблицы} WHERE
{условие};
```

На служебные слова затрачивается 18 символов (включая пробел). Атрибуты могут иметь вариативные названия, но чаще всего их стараются сделать как можно короче. В случае таблицы исследуемой ИС нужные для работы атрибуты имеют длину названия порядка 4 символов. Нам нужен атрибут, содержащий текст сообщения и id записи:

| **[SELECT ]**id, text [*8 символов*] |
|---|

К длине запроса добавится (8 + 1) символов. 1 дополнительный символ идёт на разделительный знак пробела.

Название таблицы в используемой СУБД имеет размер 5 символов: добавляем (5+1) символов к длине запроса. Условие накладывается на дату получения сообщения и имеет общий вид:

| **[WHERE ]**date between ‘YYYY-MM-DD’ and ‘YYYY-MM-DD’; [*43 символа*] |
|---|

Итого получаем в среднем 18+(4+1)+(5+1)+43 = 72 символа.

$m_{sup}$ – количество символов в вспомогательной информации для подключения к СУБД: имя пользователя (8 символов), пароль пользователя (8 символов) и номер порта (4 символа). $m_{sup} = 8 * 2 + 4 = 20$ (символов)

В СУБД PostgreSQL используется по умолчанию кодировка UTF-8. В запросе не используется кроме латиницы дополнительных символов, за исключением чисел, кавычек и пробела, а следовательно все символы лежат в диапазоне (0000—007F) Юникода, для их кодирования необходим один октет, $N$ = 256.

В соответствии с протоколом IPv4 IP-адрес кодируется 32 бит. Пакет содержит IP-адрес получателя, отправителя, 128 бит дополнительной информации, определяющей правильный путь пакета в сети, и собственно данные пользователя. Тогда значение $i_{id} = 32 * 2 + 128 = 192$ (бит).

Скорость передачи информации $V$ имеет большой диапазон вариативности в зависимости от положения точки доступа, времени суток и множества других параметров, поэтому для оценки этого параметра воспользуемся данными о качестве связи исследуемой информационной системы. Скорость передачи данных $V \approx 92$ Мбит/с.

Исходя из устройства таблицы в базе данных исследуемой системы в день приходит по 10 писем, для исследования отбираются письма за период 3 дней. Таким образом $k = 10 * 3 = 30$.

Рассмотрим сумму $\sum_a [Avg(m_a) \cdot \lfloor \log_2 N_a \rfloor]$.

Поскольку мы получаем запись только со значениями id (атрибут 0) и text (атрибут 1), параметр *a* принимает значения [0, 1].

В СУБД PostgreSQL согласно технической документации одна запись в поле формата `date` и в поле формата `serial` фиксировано занимает 4 байта памяти базы данных. В поле формата `text` запись занимает количество информации, вычисляемое по уже приведённой формуле, и пропорционально количеству символов, с учётом количества символов в алфавите, которое зависит от кодировки. Как уже говорилось используется кодировка UTF-8, а диапазон значений символов, используемых в письмах вновь не выходит за интервал (0000—007F) Юникода.

Тогда:

$$Avg(m_a) \cdot \lfloor \log_2 N_a \rfloor = \begin{cases} 4, & a = 0 \\ Avg(m_a) \cdot \lfloor \log_2 256 \rfloor = Avg(m_a) \cdot 8, & a = 1 \end{cases} \quad (22)$$

Определим теперь значение $Avg(m_a)$. Для этого было определено среднее арифметическое значение количества символов в текстах, содержащихся в базе нашей исследуемой системы: $Avg(m_a) \approx 1082{,}92$ символа.

Скорость передачи данных $V \approx 92$ Мбит/с $= 96468992$ бит/с.

Оценка времени передачи по каналам:

$$t_{\text{канала}_1} = \frac{(72 + 20) \cdot \lfloor \log_2 256 \rfloor + 192}{96468992} = 1 \times 10^{-5}(\text{с}) = 0{,}01(\text{мс})$$

$$t_{\text{канала}_2} = \frac{30 \times (4 \cdot 8 + 1082{,}92 \cdot 8) + 192}{96468992} = 2{,}7 \times 10^{-3}(\text{с}) = 2{,}7(\text{мс})$$

Далее переходим к оценке времени передачи запроса.

Данный параметр закономерно зависит от количества записей в таблице базы данных, поскольку считывание данных происходит путём прохода по всем записям таблицы, проверки выполнения накладываемого условия (если оно есть) и помещения подходящих записей в память – затем полученный результат выдаётся человеку-оператору СУБД. Чтобы оценить эту характеристику, обратимся к уже существующим исследованиям зависимости обработки времени запроса от количества записей в базе данных.

Оценим время выполнения запроса в СУБД. Поскольку мы не производим объединения таблиц и осуществляем простое чтение запросом типа SELECT формула приобретает вид:

$$t_{\text{запроса}} = \sum_{i=0}^{3} (\varphi_i \cdot t_i) = 1 \cdot t_0 + 0 \cdot t_1 + 0 \cdot t_2 + 0 \cdot t_3 = t_0$$

В литературе указано, что для СУБД PostgreSQL при количестве записей 1000 (что соответствует характеристике исследуемой системы) время обработки простого запроса на чтение без объединения таблиц составляет $t_{\text{запроса}} = t_0 = 87{,}3\ (\text{мс}) = 0{,}0873\ (\text{с})$.

К сожалению данные о сочетании архитектуры word2vec и KNN в литературе не описаны, но для алгоритма векторизации word2vec мы имеем данные о его обучении совместно с архитектурой TCN (Temporal Convolutional Network), которая в данном случае отвечает за задачу классификации. Поскольку алгоритм классификации k-ближайших соседей проходит только один раз, то возьмём в качестве аналога одну эпоху (проход по всей обучающей выборке) обучения word2vec+TCN.

Приблизительная оценка, которая может быть дана времени обучения в исследуемой системе тогда: $t_{\text{обучения}} \approx 17{,}4\ (\text{с})$

Время работы кода можно примерно определить, зная скорость загрузки данных в среду и количество информации в файлах импортируемых библиотек и обучающих данных. Скорость загрузки данных $v = 45\ (\text{МБ/с})$. Файл csv с обучающими данными имеет размер $I_{learn} = 5.5$ МБ, файлы импортируемых библиотек суммарно имеют размер около $I_{lib} = 62.5$ МБ.

Воспользовавшись формулой передачи данных по каналу (но в данном случае канал внутренний по шине) и формулой времени выполнения кода, получаем следующую формулу времени:

$$t_{\text{кода}} = \frac{I_{learn} + I_{lib}}{v} \tag{23}$$

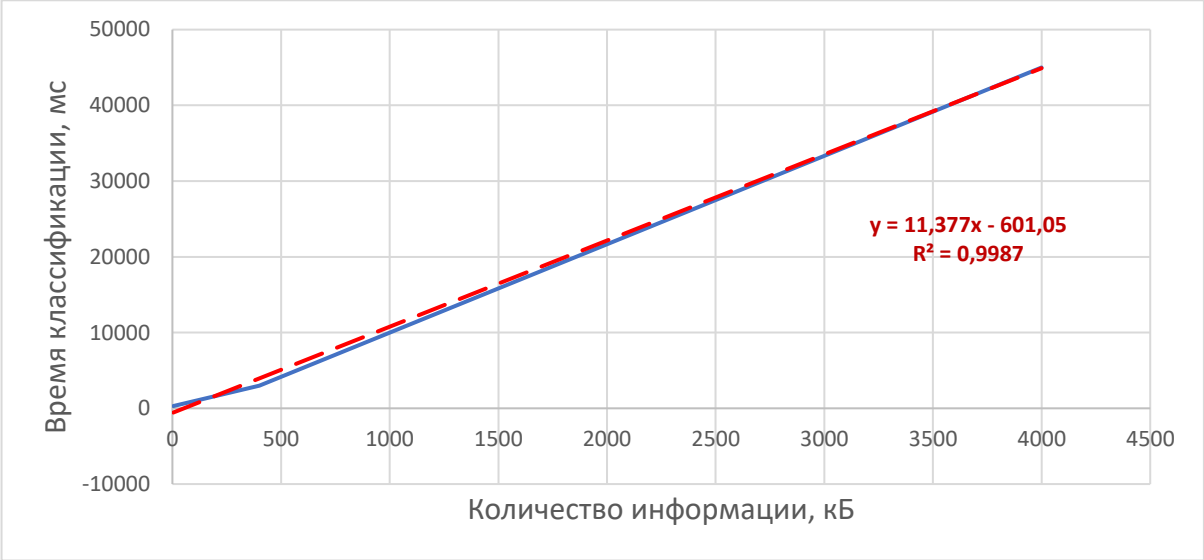
Подставив значения, получим:

$$t_{\text{кода}} = \frac{5.5 + 62.5}{45} = 1{,}511(\text{с})$$

Для алгоритма классификации k-ближайших соседей, KNN, в литературе приводятся следующие данные:

| Количество информации (кБ) | Скорость классификации (мс) |
|---|---|
| 4,00 | 300,00 |
| 400,00 | 3000,00 |
| 4000,00 | 45000,00 |

Построим по этим данным график и попробуем аппроксимировать его линейной функцией:




Насколько можно судить по трём измерениям, приближение (обозначенное красным пунктиром) можно назвать удачным. Воспользуемся полученным уравнением приближения. В данном случае выражение для времени классификации будет таким:

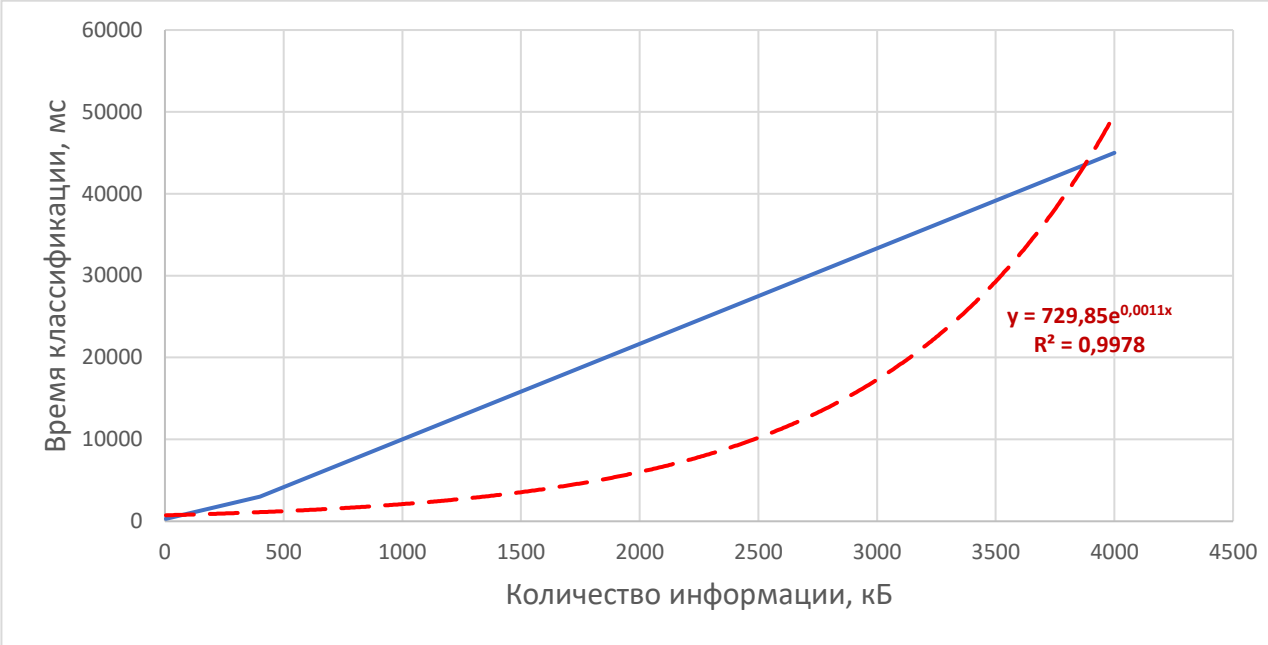
$$t_{\text{классиф.}}(k, m_a) = 11{,}377 \cdot k \cdot \frac{(Avg(m_1) \cdot \log_2 N_1)}{8000 \text{ бит}} - 601{,}05$$

Мы уже оценивали данные параметры ранее: $k$ = 30; $Avg(m_1)$ = 1082,92; $\log_2 N_1$ = 8.

$$t_{\text{классиф.}} = 11{,}377 \cdot 30 \cdot \frac{1082{,}92 \cdot 8}{8000} - 601{,}05 = -231.44(\text{мс})$$

Тут мы попали в неблагоприятную для наших значений область (количество информации в сообщениях около 32 кБ), где тренд заходит в отрицательную область.

Воспользуемся экспоненциальной аппроксимацией:




Экспоненциальная оценка тоже хорошо описывает данные. Тогда:

$$t_{\text{классиф.}}(k, m_a) = 729{,}85 \cdot e^{\left(0{,}0011 \cdot k \cdot \frac{(Avg(m_1) \cdot \log_2 N_1)}{8000 \text{ бит}}\right)}$$

Вычисляем аналогично:

$$t_{\text{классиф.}} = 729{,}85 \cdot e^{\left(0{,}0011 \cdot 30 \cdot \frac{(1082{,}92 \cdot 8)}{8000 \text{ бит}}\right)} \approx 756(\text{мс})$$

| Параметр времени | Полученное значение (с.) |
|---|---|
| $t_{\text{обучения}}$ | 17,4 |
| $t_{\text{кода}}$ | 1,511 |
| $t_{\text{запроса}}$ | 0,0873 |
| $t_{\text{канала}_1}$ | 0,00001 |
| $t_{\text{канала}_2}$ | 0,0027 |
| $t_{\text{классиф.}}$ | 0,756 |
| **Общее время:** | 19,75701 |

***Результаты экспериментального измерения времени***

После запуска сценария второй исследуемой системы МО получаем следующие результаты для временных характеристик в соответствии с формулой (21):

| Параметр времени | Полученное значение (с.) |
|---|---|
| $t_{\text{обучения}}$ | 7,06 |
| $t_{\text{кода}}$ | 1,515 |
| $t_{\text{запроса}}$ | 0,017 |
| $t_{\text{канала}_1}$ | 0,000009 |
| $t_{\text{канала}_2}$ | 0,00275 |

| | |
|---|---|
| $t_{\text{классиф.}}$ | 0,0625 |
| **Общее время:** | 8,657259 |

Получено значение точности accuracy для первого класса: 0.9741379310344828; и для класса «спам»: 0.9

Как видно, модель достаточно неплохо справляется с задачей, а значит мы хорошо её обучили.

***Анализ результатов***

*Сравнение теоретических и практических данных*

Для системы машинного обучения:

| Параметр времени | Теоретические данные | Экспериментальные данные |
|---|---|---|
| $t_{\text{обучения}}$ | 17,4 | 7,06 |
| $t_{\text{кода}}$ | 1,511 | 1,515 |
| $t_{\text{запроса}}$ | 0,0873 | 0,0170 |
| $t_{\text{канала}_1}$ | 0,00001 | 0,000009 |
| $t_{\text{канала}_2}$ | 0,0027 | 0,00275 |
| $t_{\text{классиф.}}$ | 0,756 | 0,0625 |
| **Общее время:** | 19,75701 | 8,657259 |

Здесь мы наблюдаем большие колебания в показателях. Тем не менее, стоит отметить, что оценка времени передачи по каналу и времени выполнения кода имеют высокую точность оценки.

Колебание во времени обучения, скорее всего, обусловлено необходимостью проводить приблизительную оценку по аналогам, которые имеют не точное соответствие исследуемой системе. Эта характеристика имеет зависимость от количества данных и выбранной модели, но эту связь трудно адекватно описать по аналогам. Тем не менее, учитывая разброс времени обучения в разных моделях, мы можем получить примерную оценку порядка параметра времени обучения модели, экспериментальные и теоретические данные имеют схожий порядок около $10^1$ с.

Получено меньшее время выполнения запроса на чтение для кода на Python. Возможно, это связано с тем, что СУБД тратит значительное время на процесс создания данных в нужном виде для дальнейшего отображения экранных форм ЧО. В коде Python такой необходимости нет. Соответственно, они отличаются на 70 мс.

Для оценки времени классификации получены экспериментальные данные, отличающиеся на порядок друг от друга. Отчасти причиной этого может быть проблема в наличии всего трёх временных точек для приближения, а также различия в моделях: в случае теоретической оценки мы имеем 4 класса, а в исследуемой системе их всего 2. К тому же на вход KNN подаётся предварительно обработанная информация в виде векторов, что может упрощать процесс классификации.

В целом, в разработанной оценке имеются характеристики, которые дают не совсем адекватное представление о модели системы, однако, если имеет место возможность ошибки в пределах одного порядка, то данная оценка может быть эффективна. Для сложных систем машинного обучения, где время обучения составляет гораздо большие значения, ошибка на порядок может оказаться не столь существенна.

## Заключение.

Исследованы методы работы с мультимодальными данными, графами и текстом. Проведён анализ визуализации данных в графовом видес помощью библиотеки igraph. Проведено исследование модели векторизации на текстовых данных и эргономичность данной системы машинного обучения.

Система оценки параметра времени для машинного обучения показала наличие большой размах ошибки, тем не менее её можно использовать для приблизительной оценки эргономичности сложных моделей, где отличие значения теоретического и практического на порядок могут быть не столь критичны.

## Список литературы


1. Biswas R. Embedding based link prediction for knowledge graph completion [Text] – In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020 – pp. 3221-3224.
2. Bordes A. Translating embeddings for modeling multi-relational data [Text]/ Bordes A., Usunier N., Garcia-Duran A., Weston J., Yakhnenko O. – Advances in neural information processing systems, 2013. – p. 26.
3. De Vries G. K. D. A Fast and Simple Graph Kernel for RDF [Text]/ De Vries G. K. D., De Rooij S. – DmoLD, 2013 – p. 1082.
4. Ji S. A survey on knowledge graphs: Representation, acquisition, and applications [Text]/ Ji S., Pan S., Cambria E., Marttinen P., Philip S. Y. – IEEE Transactions on Neural Networks and Learning Systems, 2021.
5. Kannan A. V. Multimodal knowledge graph for deep learning papers and code [Text]/ Kannan A. V., Fradkin D., Akrotirianakis I., Kulahcioglu T., Canedo A., Roy A., Al Faruque M. A. – In Proceedings of the 29th ACM International Conference on Information & Knowledge Management, 2020, October. – pp. 3417-3420.
6. Lin Y. Multimodal Learning on Graphs for Disease Relation Extraction [Text]: preprint arXiv:2203.08893/ Lin Y., Lu K., Yu S., Cai T., Zitnik M. – arXiv, 2022.
7. Ramachandram D. Deep multimodal learning: A survey on recent advances and trends [Text]/ Ramachandram D., Taylor G. W. – IEEE signal processing magazine, 34(6), 2017 – pp. 96-108.
8. Sap M. et al. Atomic: An atlas of machine commonsense for if-then reasoning //Proceedings of the AAAI conference on artificial intelligence. – 2019. – Т. 33. – №. 01. – С. 3027-3035.
9. Sun J. et al. Categorizing malware via a Word2Vec-based temporal convolutional network scheme //Journal of Cloud Computing. – 2020. – Т. 9. – №. 1. – С. 1-14.
10. Trstenjak B., Mikac S., Donko D. KNN with TF-IDF based framework for text categorization //Procedia Engineering. – 2014. – Т. 69. – С. 1356-1364.

11. Wang Q. Knowledge graph embedding: A survey of approaches and applications [Text]/ Wang Q., Mao Z., Wang B., Guo L. – IEEE Transactions on Knowledge and Data Engineering, 29(12), 2017. – pp. 2724-2743.

12. Wang Y. Fake news detection via knowledge-driven multimodal graph convolutional networks [Text]/ Wang Y., Qian S., Hu J., Fang Q., Xu C.// In Proceedings of the 2020 International Conference on Multimedia Retrieval. – 2020. – pp. 540-547.

13. Wang Z. Multimodal data enhanced representation learning for knowledge graphs [Text]/ Wang Z., Li L., Li Q., Zeng D.//In 2019 International Joint Conference on Neural Networks (IJCNN) – IEEE, 2019, June. – pp. 1-8.

14. Wilcke W. X. (2020). End-to-End Entity Classification on Multimodal Knowledge Graphs [Text]/ Wilcke W. X., Bloem P., de Boer V., van't Veer R. H., van Harmelen F. A. H. – arXiv, 2020.

15. Wu X. Multimodal news story clustering with pairwise visual near-duplicate constraint [Text]/ Wu X., Ngo C. W., Hauptmann A. G. – IEEE Transactions on Multimedia, 10(2), 2008. – pp.188-199.

16. Горячкин Б. С. Эргономический анализ систем обработки информации и управления //Вестник евразийской науки. – 2017. – Т. 9. – №. 3 (40). – С. 72.

17. Елисеева Е. А., Горячкин Б. С., Виноградова М. В. ИССЛЕДОВАНИЕ ПРОИЗВОДИТЕЛЬНОСТИ СУБД ПРИ РАБОТЕ С КЛАСТЕРНЫМИ БАЗАМИ ДАННЫХ НА ОСНОВЕ ЭРГОНОМИЧЕСКОГО АНАЛИЗА //StudNet. – 2022. – Т. 5. – №. 4. – С. 2888-2910.

18. [Электронный ресурс] igrhttps://igraph.org/aph – Network analysis software – документация по библиотеке визуализации графов igraph.