МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСКОГО»

Кафедра информатики и программирования

# РАСПАРАЛЛЕЛИВАНИЕ ЗАДАЧИ ОБРАБОТКИ ТЕКСТА НА ЕСТЕСТВЕННОМ ЯЗЫКЕ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы
направления 02.03.03 Математическое обеспечение и администрирование информационных систем
факультета компьютерных наук и информационных технологий
Шаталов Даниил Родионович

Научный руководитель:
старший преподаватель _____ М.С. Портенко

подпись, дата

Зав. кафедрой:
к.ф.-м.н., доцент _____ М.В. Огнева

подпись, дата

Саратов 2021

# ВВЕДЕНИЕ

**Актуальность темы.** Довольно часто компьютерные программы сталкиваются с длительными процессами. Например, это могут быть сложные математические вычисления или обработка объемных данных. Пока такие процессы выполняются необходимо ждать их завершения, а сопутствующее этому ожиданию бездействие системы может приводить к крупным убыткам и быстрому снижению продуктивности. Такое выполнение программы, подразумевающее последовательное выполнение операций, называется синхронным. Наиболее эффективно было бы распределить время работы программы следующим образом: во время выполнения некоторой сложной операции, происходила бы обработка еще нескольких, от которых первая не зависит. Такой подход называется асинхронным программированием. Оно позволяет увеличивать эффективность рабочего процесса, благодаря тому что не разрешает блокировать основной поток выполнения. Асинхронность говорит о порядке исполнения кода, то есть, в случае если вызов функции не дает результата сразу, но отдаёт управление вызывающему ее коду с сообщением вернуть результат позднее, то такая функция будет являться асинхронной.

Также говоря об асинхронности, нельзя не упомянуть о связанных с ней не менее важных понятиях. Это такие понятия как конкурентность (concurrency), параллелизм (parallel execution) и многопоточность (multithreading). Все это связано с одновременным выполнением задач, но под каждым термином подразумеваются свои особенности.

Параллелизм подразумевает, что физически происходит выполнение нескольких задач одновременно. Он используется для разделения единой задачи на подзадачи, чтобы добиться ускорения производимых вычислений. При этом наличие нескольких физических потоков в данном случае очень важно из-за того, что на компьютере с одним вычислительным устройством (процессорным ядром) данную процедуру невозможно произвести.

Конкурентность – это возможность разбивать программу на отдельные части, выполняющиеся независимо и коммуницирующие между собой. Другими словами, это возможность выполнения множества задач одновременно. Результат выполнения конкурентной программы будет таким же, как и при последовательном выполнении, но преимущество конкурентности заключается в том, что те же результаты будут получены за меньшее время, что только улучшит производительность. Конкурентность вытекает из подхода к структурированию логики выполнения программы, то есть ее деления на части, выполняющиеся параллельно на нескольких ядрах или же поочередно на одном. Таким образом, конкурентность близка к проектированию программного обеспечения, а параллелизм к способу его выполнения.

В многопоточности поток является абстракцией, под которым может подразумеваться и отдельное ядро процессора, и поток выполнения операционной системы. В некоторых языках программирования для реализации многопоточности имеются свои собственные объекты потоков. Поэтому такой подход может иметь принципиально различную реализацию.

Таким образом, процесс распараллеливания заключается в адаптации программ для достижения их эффективного исполнения на вычислительной системе параллельной архитектуры. На сегодняшний день чаще всего распараллеливание происходит именно на многопроцессорной вычислительной системе.

На сегодняшний день параллельные вычисления – это современная и многогранная область, которая быстро развивается и является наиболее актуальной в ближайшие десятилетия. Актуальность данной области складывается из множества факторов, но одним из самых главных является потребность в огромных вычислительных ресурсах для решения сложных прикладных задач моделирования процессов в физике, биофизике, химии и других областях. В данной работе рассматривается применение распараллеливания вычислений в области обработки естественных языков.

**Цель бакалаврской работы** – реализация алгоритма классификации текста и его распараллеливание.

Поставленная цель определила **следующие задачи**:

1. изучение теоретических основ обработки естественного языка;
2. изучение способов решения задачи классификации текста;
3. анализ инструментов для построения параллельных приложений на платформе JVM;
4. выбор и разработка алгоритма для решения задачи классификации текста;
5. выполнение предобработки текстовых данных для обучения выбранного алгоритма;
6. распараллеливание алгоритма средствами языка Java;
7. распараллеливание алгоритма с помощью фреймворка Akka;
8. анализ и сравнение полученных результатов времени выполнения программ.

**Методологические основы** распараллеливания задачи обработки текста на естественном языке представлены в работах Л. Хобсона, В. А. Яцко, К. Богачева, И. Федотова, П. Батчера, Р. Рестенбурга.

**Практическая значимость бакалаврской работы.**

Созданный алгоритм может быть использован в системах, где есть необходимость в быстрой классификации текста. Примером могут послужить системы, которые отвечают за аукцион рекламных объявлений в реальном времени.

**Структура и объём работы.** Бакалаврская работа состоит из введения, 6 разделов, заключения, списка использованных источников и 4 приложений. Общий объем работы – 70 страниц, из них 54 страниц – основное содержание, включая 52 рисунка и 1 таблицу, список использованных источников информации – 24 наименования.

## КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

**Первый раздел «Постановка задачи»** посвящен обоснованию актуальности задачи распараллеливания алгоритма обработки текста на естественном языке.

**Второй раздел «Обработка текстов на естественном языке»** посвящен описанию способов обработки текса, среди которых имеются следующие: классификация текста, токенизация текста, лемматизация слов, векторизация текста, дедупликация данных.

Обработка текстов на естественном языке (Natural Language Processing, NLP) – общее названия для направления искусственного интеллекта и математической лингвистики, включающее большой спектр задач различного уровня. В задачи этого направления входит изучение проблем компьютерного анализа, а также синтеза текстов на естественных языках. Если рассматривать область искусственного интеллекта, то здесь анализ подразумевает понимание самого языка, а синтез будет подразумевать генерацию грамотно составленного текста.

Классификация текстов (документов) – это задача компьютерной лингвистики, цель которой состоит в отнесении документа к некоторой категории на основании содержания документа. Данная задача может применяться для борьбы со спамом, определении языка, выбора наиболее уместной рекламы. Сюда же входит и разделение сайтов, в том числе и новостных, по тематическим разделам.

Токенизацией является процесс, заключающийся в сегментации текста на некоторые единицы.

Лемматизация – это этап приведения слова к его смысловой канонической форме, являющийся важным шагом при предварительной обработке текста. Данный алгоритм часто используется в случаях обработки естественного языка и других лингвистических областях.

Под векторизацией понимается процесс конвертирования слов в числа. Входными данными для большинство математических моделей являются вектора, поэтому следует провести процедуру по векторизации, то есть получить векторное представление слов.

При дедупликации данных происходит устранение избыточных копий анализируемой информации. Один из способов проведения дедупликации –

использование локально-чувствительного хэширования. Понятие locality-sensitive hashing (LSH) или в переводе – локально-чувствительное хэширование впервые было употреблено в 1998 году, в качестве названия рандомизированной структуры хеширования для эффективного поиска ближайшего соседа в многомерном пространстве.

В отличие от обычного хэширования, локально-чувствительное хэширование имеет другую концепцию. Классический подход к хэшированию подразумевает, что пара входных значений будет схожа, но при этом не будет идентична и оба значения будут на выходе давать кардинально отличные друг от друга результаты. В основе локально-чувствительного хэширования лежит другая идея, заключающаяся в том, что схожие входные значения будут выдавать схожие выходы, у которых будет существенно меньшая размерность. LSH предоставляет инструменты для построения структуры быстрого вероятностного поиска $n$-мерных векторов, которые будут схожи с исходным правилом. Имеет смысл прибегнуть к такому подходу, когда задача содержит в себе огромные объемы данных, которые необходимо сравнивать.

**Третий раздел «Машинное обучение в задаче классификации текста»** посвящен описанию алгоритму k – ближайших соседей.

Основными шагами по работе с данным алгоритмом будут:

Шаг 1 – загрузка данных для обучения, а также данных для тестирования.

Шаг 2 – выбор значения k.

Шаг 3 – для каждой точки в тестовых данных провести следующую процедуру, состоящую из нескольких шагов:

1) Рассчитать расстояние между тестовыми данными и каждой строкой данных тренировки с помощью любой из метрик.

2) Основываясь на значении расстояния, отсортировать их в порядке возрастания.

3) Далее выбираются верхние k строк из отсортированного массива.

4) Назначается класс контрольной точке на основе наиболее часто встречающегося класса этих строк.

**Четвертый раздел «Теория параллельного программирования»** посвящен теоретическим основам параллельного программирования. Описанию принципов работы акторной модели и особенностей фреймворка Akka, имплементирующий данную модель.

Данный подход позволяет использовать несколько потоков, которые не будут конкурировать за используемые ресурсы. При этом взаимодействие между потоками является асинхронным благодаря общению основанном на пересылке сообщений. Например, взаимодействие может быть следующим.

- поток X посылает сообщение-запрос потоку Y;
- поток Y когда-нибудь получит запрос потока X и отошлет потоку X ответ;
- поток Y отсылает сообщение-обновление потоку Z;
- поток Z когда-нибудь получит сообщение-обновление от потока Y и обновит те данные, которые принадлежат потоку Z.

У каждого из потоков существует собственная очередь входящих сообщений. Любой поток берет сообщения из своей очереди и обрабатывает их. Если же очередь пуста, то поток бездействует до того, как поступят новые сообщения. Если же потоку X нужно что-то от другого потока Y, то поток X помещает сообщение во входящую очередь потока Y.

**Пятый раздел «Предварительная обработка обучающих данных»** посвящен реализации подготовки данных для последующей работы с ними.

В данном разделе были представлены python-скрипты, отвечающие за балансировку и валидацию данных в исходных выборках.

**Шестой раздел «Алгоритм kNN для решения NLP-задачи»** посвящен реализации алгоритма на языке программирования Java и последующее распараллеливание его с помощью Stream API и фреймворка Akka.

В данном разделе была представлена реализации следующие алгоритмы: алгоритм kNN, алгоритм токенизации текста, алгоритм лемматизации слов, использующий словарь лемм и алгоритм вычисления редакционного расстояния или расстояния Левенштейна, алгоритм, вычисляющий TF-IDF, алгоритм, использующий локально-чувствительное хеширование.

В ходе работы были получены следующие результаты, отображенные в таблице 1.

Под номерами понимаются алгоритмы, где №1 – синхронный и не распараллеленный, №2 – синхронный и распараллеленный, а №3 – асинхронный и распараллеленный. Во всех алгоритмах использовались две приблизительно равные по весу (Мб) обучающие выборки с разных интернет-ресурсов: «Яндекс.Новости» и «Lenta.ru». Однако у выборки с «Яндекс.Новсти» большее количество новостей, но объем самих новостей меньше, чем у выборки с «Lenta.ru».

Таблица 1 – Время работы алгоритмов на данных с ресурса «Яндекс.Новости» и с ресурса «Lenta.ru»

| | Яндекс.Новости (9мб) | | | Lenta.ru (10мб) | | |
|---|---|---|---|---|---|---|
| Рассматриваемый алгоритм | №1 | №2 | №3 | №1 | №2 | №3 |
| Лемматизация тренировочной выборки (с.) | 757 | 360 | 454 | 688 | 490 | 569 |
| Вычисление TF-IDF для тренировочной выборки (с.) | 1370 | 183 | 59 | 2278 | 298 | 373 |
| Процесс векторизации для тренировочной выборки (с.) | 21 | 5 | 7 | 8 | 3 | 30 |
| Вычисление LSH для тренировочной выборки (с.) | 486 | 70 | 87 | 176 | 30 | 32 |
| Лемматизация тестовой выборки (с.) | 9 | 1 | 31 | 3 | 1 | 47 |
| Вычисление TF-IDF для тестовой выборки (с.) | 1 | 1 | 11 | 1 | 1 | 33 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Процесс векторизации для тестовой выборки (с.) | 3 | 1 | 1 | 1 | 1 | 1 |
| Вычисление LSH для тестовой выборки (с.) | 4 | 9 | 10 | 1 | 2 | 1 |
| kNN (с.) | 66 | 27 | 5 | 19 | 2 | 1 |
| Итого (с.) | 2717 | 645 | 640 | 3175 | 827 | 701 |

На обеих выборках первый алгоритм оказался самым медленным. А второй и третий алгоритм практически равны на выборке «Яндекс.Новости», но на данных с ресурса «Lenta.ru» третий алгоритм работает на 15% быстрее, чем второй.

## ЗАКЛЮЧЕНИЕ

Таким образом, поставленная в работе цель, заключающаяся в распараллеливании алгоритма, для решения задачи классификации текста была достигнута.

Для достижения данной цели были решены все следующие задачи:

- был выбран, проанализирован и разработан алгоритм для решения задачи классификации текста;
- произведена процедура предобработки данных, для более удобной работы с ними в программном коде;
- создана процедура для предобработки полученных текстовых данных для задачи классификации текста;
- реализован синхронный и не параллельный алгоритм kNN для решения задачи классификации текста;
- используя Stream API, процедура предобработки данных и алгоритм для классификации были распараллелены;
- также процедура предобработки данных и алгоритм для классификации были распараллелены другим способом – с помощью фреймворка Akka.

Можно сделать вывод, что параллельный и асинхронный код показал себя лучше остальных в данной задаче, так как выполнялся быстрее. За ним

на втором месте следует параллельный и синхронный код. Третье место занимает не параллельный и синхронный.

Есть возможность в дальнейшем добавить в алгоритм метод стемматизации слов, улучшить векторизацию используя подход Word2Vec.

**Основные источники информации:**


1. Хобсон, Л. Обработка естественного языка в действии / Л. Хобсон, Х. Ханнес, Х. Коул - СПб. : Питер, 2020. - 576 с.
2. Яцко, В.А. Алгоритмы и программы автоматической обработки текста / В.А. Яцко - Вестник ИГЛУ. 2012. - 150 с.
3. Богачев, К. Программирование. Основы параллельного программирования / К. Богачев - М. : Бином. Лаборатория знаний, 2015. - 344 с.
4. Федотов, И. Модели параллельного программирования / Федотов И. - М. : СОЛОН-Пресс, 2017. - 392 с.
5. Батчер, П. Семь моделей конкуренции и параллелизма за семь недель / П. Батчер : пер. А. Киселев. - М. : ДМК-Пресс, 2015. - 360 с.
6. Рестенбург, Р. Akka в действии / Р. Рестенбург, Р. Уильямс, Р. Баккер : пер. А. Киселев - М. : ДМК-Пресс, 2019. - 522 с.