**Григорьев Ю.А., Пьянзин С.А.**

**Методические указания к лабораторной работе 5.**

**Работа с документной NoSQL БД** **Elasticsearch и реализация процесса data science**

Оглавление

[1. Цель и задачи лабораторной работы 1](#_Toc25922275)

[1.1. Краткая характеристика объекта изучения, исследования 1](#_Toc25922276)

[1.1.1. Особенности 2](#_Toc25922277)

[1.1.2. Применение 2](#_Toc25922278)

[1.2. Общая задача лабораторной работы 3](#_Toc25922279)

[2. Схема и описание лабораторной установки 3](#_Toc25922280)

[2.1. Установите и запустите Elasticsearch 3](#_Toc25922281)

[3. Реализация процесса Data Science на примере диагностики болезней 5](#_Toc25922282)

[3.1. Сбор и подготовка данных 6](#_Toc25922283)

[3.2. Исследование данных 7](#_Toc25922284)

[4. Профилирование болезни 9](#_Toc25922285)

[5. Примеры поиска документов в Elasticsearch с использованием виртуальной среды Python 10](#_Toc25922286)

[6. Содержание отчета по лабораторной работе 11](#_Toc25922287)

[7. Источники информации 11](#_Toc25922288)

Внимание! Перед выполнением лабораторной работы необходимо полностью прочитать методические указания.

# Цель и задачи лабораторной работы

Целью пятой лабораторной работы является формирование следующей компетенции: студент должен изучить модель представления данных и способы работы с документной NoSQL базой данных Elasticsearch. Получить навыки инсталляции, индексации и поиска в Elasticsearch, реализовать этапы процесса Data Science, протестировать работу с большим объёмом данных.

## Краткая характеристика объекта изучения, исследования

**Elasticsearh** (в дальнейшем для сокращения может обозначаться как "ES") – это масштабируемый полнотекстовый поисковый и аналитический движок с открытым исходным кодом. Он позволяет хранить большие объемы данных, проводить среди них быстрый поиск и аналитику (почти в режиме реального времени). Как правило, он используется в качестве базового механизма/технологии, которая обеспечивает работу приложений со сложными функциями и требованиями к поиску. Elasticsearch использует в качестве поисковой основы библиотеку Lucene, которая написана на языке Java и доступна для многих платформ. Все неструктурированные данные хранятся в формате JSON, что автоматически делает ES базой данных NoSQL. Но в отличие от других баз данных NoSQL, ES предоставляет возможности поиска и многие другие функции.

### Особенности

* Масштабируемость и отказоустойчивость — Elasticsearch легко масштабируется. К уже имеющейся системе можно на ходу добавлять новые серверы, и поисковый движок сможет сам распределить на них нагрузку. При этом данные будут распределены таким образом, что при отказе какого-то из узлов они не будут утеряны и сама поисковая система продолжит работу без сбоев.
* Мультиарендность (англ. multitenancy) — возможность организовать несколько различных поисковых систем в рамках одного объекта Elasticsearch. Организацию можно проводить динамически.
* Операционная стабильность — на каждое изменение данных в хранилище ведется логирование сразу на нескольких ячейках кластера для повышения отказоустойчивости и сохранности данных в случае разного рода сбоев.
* Отсутствие схемы (англ. schema-free) — Elasticsearch позволяет загружать в него обычный JSON-объект. Далее индексация и добавление в базу поиска производится автоматически. Таким образом, обеспечивается быстрое прототипирование.
* RESTful API — Elasticsearch практически полностью управляется по HTTP с помощью запросов в формате JSON.

### Применение

* Вы запускаете интернет-магазин, где вы позволяете своим клиентам искать продукты, которые вы продаете. В этом случае Elasticsearch можно использовать для хранения всего каталога продуктов и инвентаря, а также для поиска и автозаполнения предложений по ним.
* Вы хотите собирать данные журналов или транзакций и анализировать эти данные для поиска тенденций, статистики, обобщений или аномалий. В этом случае можно использовать Logstash (часть стека Elasticsearch/Logstash/Kibana) для сбора, агрегирования и анализа данных, а затем использовать Logstash для передачи этих данных в Elasticsearch. После того, как данные находятся в Elasticsearch, вы можете запускать поиск и агрегирование для получения любой интересующей вас информации.
* Вы запускаете платформу оповещения о цене, которая позволяет подкованным в цене клиентам указать правило вроде "Я заинтересован в покупке конкретного электронного гаджета, и я хочу получать уведомления, если цена гаджета упадет ниже $X от любого поставщика в течение следующего месяца". В этом случае вы можете соскоблить цены поставщиков, подтолкнуть их в Elasticsearch и использовать его возможность обратного поиска (перколятор), чтобы соответствовать движениям цен на запросы клиентов и в конечном итоге вытолкнуть оповещения клиенту, как только совпадения будут найдены.
* У вас есть потребности в аналитике / BI, и вы хотите быстро исследовать, анализировать, визуализировать и задавать специальные вопросы по большому количеству данных (речь идёт о миллионах или миллиардах записей). В этом случае можно использовать Elasticsearch для хранения данных, а затем использовать Kibana (часть стека Elasticsearch/Logstash/Kibana) для создания пользовательских панелей мониторинга, которые могут визуализировать важные для вас аспекты данных. Кроме того, можно использовать функцию агрегирования Elasticsearch для выполнения сложных запросов бизнес-аналитики к данным.

Официальная документация доступна по ссылке: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

## Общая задача лабораторной работы

Изучить модель представления данных и способы работы с документной NoSQL базой данных Elasticsearch. Получить навыки инсталляции, индексации и поиска в Elasticsearch, реализовать этапы процесса Data Science, протестировать работу с большим объёмом данных.

# Схема и описание лабораторной установки

1. Запустите виртуальную машину. Откройте терминал (пункт Open Terminal контекстного меню).

2. Создайте каталог lab5 в домашнем каталоге /home/user

mkdir lab5

3. Перепишите файл с описанием болезней data.json из яндекс-папки /Файлы/lab5 в /home/user/lab3 своей ВМ.

## Установите и запустите Elasticsearch

1. Создайте каталог elastic в домашнем каталоге /home/user

mkdir elastic

2. Перепишите архив elasticsearch-7.17.0-linux-x86\_64.tar.gz из яндекс-папки /Файлы/lab5 в /home/user/elastic своей ВМ.

3. Распакуйте архив

cd elastic

tar -xzf elasticsearch-7.17.0-linux-x86\_64.tar.gz

4. Удалите архив (необязательно)

rm elasticsearch-7.17.0-linux-x86\_64.tar.gz

5. Для запуска Elasticsearch используйте команду

elasticsearch-7.17.0/bin/elasticsearch

6. Проверить работу Elasticsearch можно командой

curl -X GET "localhost:9200/?pretty"

Появится что-то подобное:



## Установите клиента Elasticsearch для работы с python:

sudo apt install python3-pip

pip install Elasticsearch

# Реализация процесса Data Science на примере диагностики болезней

Вот как работает процесс Data Science:

1. Назначение цели исследования.
2. Сбор и подготовка данных – данные взяты из Википедии.
3. Исследование данных.
4. Моделирование данных (не рассматривается в данной работе).
5. Отображение результатов (не рассматривается в данной работе).

**Этап 1 - Назначение цели исследования**

Первичная цель: создание поисковой системы, которая помогала бы врачам в диагностике болезней по симптомам.

Вторичная цель: профилирование болезни – определить, какие симптомы отличают её от других болезней?

Вторичная цель полезна для образовательных задач или для формирования входных данных для нетривиальных применений.

**Этап 2 – Сбор и подготовка данных**

Сбор и подготовка данных – это в общем случае два разных этапа в процессе Data Science. Есть два источника информации: внутренние и внешние данные. В лабораторной работе в качестве внешних данных взяты и обработаны данные из Википедии (файл data.json) . Внешние данные, прочитанные из Википедии, затем следует сохранить в локальном индексе Elasticsearch.

**Этап 3 – Исследование данных**

Исследование данных является важнейшим аспектом этого примера, потому что главная цель работы представляет собой конкретный способ исследования данных по симптомам и по имени болезни.

## Сбор и подготовка данных

Elasticsearch - основная библиотека для работы с Elasticsearch на языке Python.

Используйте данный код в своем интерпретаторе:

import json

from elasticsearch import Elasticsearch

Вы будете проводить индексирование в локальном экземпляре Elasticsearch:

client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])

indexName = "medical"

# client.indices.delete(index=indexName)

client.indices.create(index=indexName)

Из предыдущих команд видно, что прежде всего необходимо создать клиента (client). При вызове Elasticsearch() для инициализации можно указать конкретный адрес.

Утверждается, что система Elasticsearch является бессхемной (schema-less); это означает, что Elasticsearch можно использовать без определения схемы базы данных и без указания информации о том, с какими данными она будет работать. И хотя в простых случаях это действительно так, в долгосрочной перспективе без схемы не обойтись. Поэтому мы создадим эту схему:

diseaseMapping = {

"properties": {

"name": {

"type": "text",

"fielddata": True

},

"title": {

"type": "text",

"fielddata": True

},

"fulltext": {

"type": "text",

"fielddata": True

}

}

}

client.indices.put\_mapping(index=indexName,

doc\_type="diseases",

include\_type\_name="true",

body=diseaseMapping)

Этот фрагмент сообщает Elasticsearch, что ваш индекс полей будет содержать тип документа с именем «diseases»; в diseaseMapping вы указываете тип каждого из полей, документ содержит три поля, все три поля имеют текстовый формат. Если бы отображение не было указано, то система Elasticsearch постаралась бы угадать их типы по первому полученному элементу. Также добавляется возможность сортировки, агрегации и доступа к полям с помощью - "fielddata": True

Fielddata – это структура данных в памяти, используемая для поддержки сортировки и агрегации текстовых полей. По умолчанию он выключен для экономии места в оперативной памяти.

Кластер ElasticSearch может содержать несколько индексов, которые, в свою очередь, содержат несколько типов. Эти типы содержат несколько документов, и каждый документ имеет определенные свойства.

Перейдем к индексированию болезней:

with open('data.json', 'r') as f:

dataStore = json.load(f)

for data in dataStore:

try:

client.index(

index=data["index"],

doc\_type=data["doc\_type"],

id=data["id"],

body=data["body"]

)

except Exception as e:

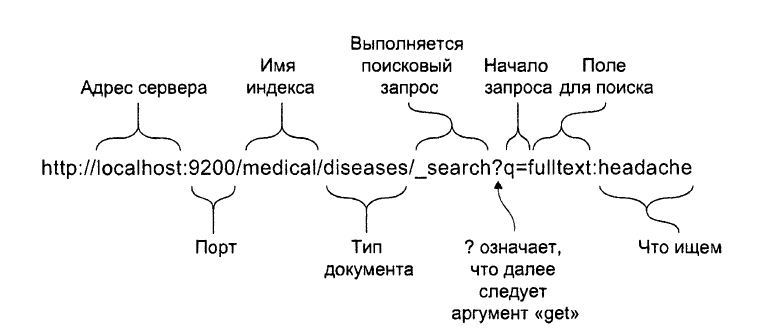
print(e)

**Примечание**. То, что выше отмечено жёлтым цветом нужно объединить в файл f1.py, сохранить его в каталоге lab5 вашей машины и запустить его с помощью команды python3:

cd lab5

python3 f1.py

Индексация займёт некоторое время, дождитесь окончания завершения программы.



Безусловно, URL-адреса Elasticsearch полезны, однако для ваших потребностей их недостаточно.

## Исследование данных

Исследование данных является важнейшим аспектом этого примера, потому что главная цель представляет собой конкретный способ исследования данных по симптомам.

В Elasticsearch поддерживается тщательно проработанный язык запросов JSON; при каждом поиске серверу отправляется запрос POST, на который сервер выдает ответ в формате JSON. В общем и целом, язык состоит из трех частей: запросы, фильтры и обобщения.

Запрос получает ключевые слова поиска и обрабатывает их одним или несколькими анализаторами, прежде чем проводить поиск этих слов в индексе. Фильтр получает ключевые слова, как и запрос, но не пытается анализировать полученную информацию; он фильтрует информацию по заданным условиям. Таким образом, фильтры обладают меньшей сложностью, но во много раз большей эффективностью, потому что они также временно сохраняются в Elasticsearch на случай повторного применения того же фильтра.

Мы воспользуемся «языком строки запроса» - способа описания запрашиваемых данных, близкого к языку поисковых запросов Google.

Простая строка **запроса 1** Elasticsearch с тремя обязательными ключевыми словами:

client = Elasticsearch([{"host": "127.0.0.1", "port": 9200}])

indexName = "medical"

docType = "diseases"

searchFrom = 0

searchSize = 3

# 1

searchBody = {

"\_source": ["name"],

"query": {

"simple\_query\_string": {

"query": '+fatigue+fever+"joint pain"',

"fields": ["fulltext", "title^5", "name^10"]

}

}

}

result = client.search(index=indexName, body=searchBody, from\_=searchFrom, size=searchSize)

При помощи "searchFrom" указывается, с какого документа начинается поиск, "searchSize" указывает на количество элементов, которые будут отображаться. В поиске возвращаются только первые три результата; по умолчанию возвращаются пять.

Начнем с трех симптомов: слабость, (fatigue), повышенная температура (fever) и суставные боли (joint pain) – см. второй "query".

В переменной searchBody, содержащей структуру данных JSON, указываются поля, которые вы хотите получить в возвращаемых результатах ("\_source"); в данном случае достаточно названия болезни name. Синтаксис строки запроса используется для поиска по всем индексируемым полям: fulltext, title и name. Добавляя знак ^, можно назначить каждому полю вес.

Если симптом встречается в заголовке, он в пять раз важнее вхождения в открытом тексте; если он встречается в самом названии, он считается в 10 раз более важным. Обратите внимание на то, как симптом суставной боли (joiпt раin) заключается в кавычки. Без кавычек cлoва joint и pain считались бы двумя разными ключевыми словами, а не одним выражением.

**Примечание**. Для выполнения запроса создайте в каталоге lab5 своей машины скрипт с расширением .py:

from elasticsearch import Elasticsearch

import pprint

тело запроса, приведённого выше

pprint.pprint(result)

Выполните запрос с помощью команды

python3 имя\_скрипта

Примерный результат запроса 1:

{'\_shards': {'failed': 0, 'skipped': 0, 'successful': 1, 'total': 1},

'hits': {'hits': [{'\_id': 'Chronic+fatigue+syndrome',

'\_index': 'medical',

'\_score': 117.35565,

'\_source': {'name': 'Chronic fatigue syndrome'},

'\_type': 'diseases'},

{'\_id': 'Chronic+fatigue+immune+dysfunction+syndrome',

'\_index': 'medical',

'\_score': 100.70878,

'\_source': {'name': 'Chronic fatigue immune dysfunction '

'syndrome'},

'\_type': 'diseases'},

{'\_id': 'Encephalomyelitis,+myalgic',

'\_index': 'medical',

'\_score': 39.607258,

'\_source': {'name': 'Encephalomyelitis, myalgic'},

'\_type': 'diseases'}],

'max\_score': 117.35565,

'total': {'relation': 'eq', 'value': 48}},

'timed\_out': False,

'took': 6}

Relation - eq указывает на тип отношения с точным числом попадания, а value указывает на количество найденных таких записей.

Took указывает на время выполнения в миллисекундах.

Первым результатом поиска стала болезнь - Синдром хронической усталости (Chronic fatigue syndrome). Волчанка (Lupus erythematosus) не входит в число первых 3 из 48 подходящих болезней. Результаты сортируются по рейтингу соответствия (переменная \_score), в нем учитывается, насколько хорошо болезнь соответствует запросу, сколько раз было найдено ключевое слово, назначенные веса и т.д.

Задания.

**Запрос 2**. Как изменятся результаты поиска болезней, если в "query" добавить новый симптом rash (сыпь)?

**Запрос 3**. Как изменятся результаты поиска болезней, если в "query" добавить ещё один симптом sensitivity to sunlight (чувствительность к солнечному свету)?

# Профилирование болезни

**Запрос 4** позволяет выявить симптомы болезней, в имени которых встречается слово diabetes (диабет)

# 4

searchBody={

"\_source": ["name"],

"query": {

"query\_string" : {

"query": "name: diabetes"

}

},

"aggregations" : {

"DiseaseKeywords" : {

"significant\_terms" : { "field" : "fulltext", "size" : 10 }

}

}

}

result=client.search(index=indexName, body=searchBody, from\_ = searchFrom, size=searchSize)

Вопросы.

1. Какие 5 симптомов диабета являются наиболее важными? Сколько документов (названий болезней, связанных с диабетом) было проанализировано?

2. Сколько времени выполнялся запрос?

Задания.

**Запрос 5**. Выявить симптомы болезней, в имени которых встречаются слова diabetes (диабет) и insipidus (несахарный).

**Запрос 6**. Выполните нечёткий поиск (fuzzy) для имени болезни diabetse (в названии болезни случайно поменяли местами две последние буквы).

# Примеры поиска документов в Elasticsearch с использованием виртуальной среды Python

Примеры поиска документов с использованием данных из пункта 3.1 и интерпретатора Python.

1. **Запрос 7** с фильтрацией:

filterQuery = {

"\_source": ["name", "title", "fulltext"],

"query": {

"constant\_score": {

"filter": {

"term": {

"title": "headache"

}

}

}

}

}

result=client.search(index="medical", body= filterQuery)

Запрос с типом "constant\_score" обрабатывает запрос фильтром, указанным в "filter" с правилом "term", и возвращает соответствующие документы. Перевести документы на русский язык (Google Переводчик).

2. **Запрос 8** с сортировкой:

sortQuery = {

"\_source": ["name", "title"],

"sort": [{"title": "desc"}]

}

result=client.search(index=indexName,body=sortQuery,from\_=searchFrom, size=searchSize)

В "\_source" указываются поля, а в "sort" правила для сортировки, в нашем случае результаты сортируются по текстовому полю "title" в обратном порядке.

3. **Запрос 9**: извлечение документа по идентификатору:

idQuery = {

"query": {

"terms": {

"\_id": ["Stomatitis ", "Alkaptonuria"]

}

}

}

result=client.search(index=indexName, body=idQuery)

В terms указывается, что в качестве идентификатора("\_id") используются значения "Stomatitis" и "Alkaptonuria".

4. Логический запрос (bool query) – см. документацию

<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

Логический запрос возвращает документы, которые соответствуют логическим комбинациям других запросов, к ним относятся: must, filter, should, must\_not; "must"– означает, что условие должно выполняться в соответствующих документах, эти условия соединяются по AND; "should"– как минимум одно из условий должно выполняться, условия соединяются по OR.

Задание.

Написать и выполнить **запрос 10** bool query, позволяющий найти описания болезней, удовлетворяющих следующим условиям:

1. "must": в поле "title" встречается слово "Chronic"

2. "filter": в тексте документов ("fulltext") встречается слово "Americans"

3. "should": в поле "title" встречается слово "leukemia" или "syndrome"

4. "must\_not": в поле "name" не должно встречаться сочетание "Epidemic encephalomyelitis"

5. Минимальное число низкочастотных терминов (для should), которые должны присутствовать, равно 1 ("minimum\_should\_match": 1).

# Содержание отчета по лабораторной работе

Отчет должен содержать:

- тексты запросов 1-10 (ссылка на номер запроса обязательна),

- результаты выполнения запросов (для полного текста документа: переведите на русский с помощью программы-переводчика, вставьте в отчёт первые 20 строк),

- пояснение результатов.

# Источники информации

1. Али Мохамед, Силен Дэви, Мейсман Арно. Основы Data Science и Big Data. Python и наука о данных. Санкт Петербург – 2016. – 336 с.

2. Elastic Elasticsearch, Национальная библиотека им. Н.Э. Баумана, ссылка: <https://ru.bmstu.wiki/Elastic_Elasticsearch>

3. Основы Elasticsearch, Habr, ссылка: <https://habr.com/ru/post/280488/>

4. Elastic Docs, ссылка: <https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>

5. <https://riptutorial.com/Download/elasticsearch.pdf>

6. <https://g-soft.info/articles/768/poluchit-spisok-indeksov-v-elasticsearch/>