ТЕХНОТРЕК

Лекция 9

Поисковые движки. Elasticsearch

Найдется всё или хотя бы что-то

Поисковые платформы

Elasticsearch

- Open source (1386 контрибьюторов)
- Масштабируемость и отказоустойчивость
- Удобный API
- Гибкие настройки
- Динамический маппинг
- Геопоиск
- CJK

Много незнакомых слов

Морфорлогия

Стемминг

Нечеткий поиск

Лемматизация

N-грамма

Стоп-слова

Elasticsearch концепты сверху

- Нода
- Кластер
- Шард
- Реплика

Elasticsearch концепты внутри

- Индекс
- Тип
- Документ
- Поле
- Отображение (mapping)
- Query DSL

Анализаторы

Цель - из входной фразы получить список токенов, которые максимально отражают ее суть

Пример анализатора

```
PUT /your-index/_settings
{
  "index": {
    "analysis": {
      "analyzer": {
        "customHTMLSnowball": {
         "type": "custom",
          "char_filter": [
            "html_strip"
          ],
          "tokenizer": "standard",
          "filter": [
            "lowercase",
            "stop",
            "snowball"
          ]
}}}}}
```

Расстояние Левенштейна

(редакционное расстояние, дистанция редактирования) — минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.

Цены операций могут зависеть от вида операции

w(a, b) — цена замены символа a на символ b
w(ε, b) — цена вставки символа b
w(a, ε) — цена удаления символа a

Частный случай задачи - Расстояние Левенштейна

w(a, a) = 0
w(a, b) = 1 при a≠b
w(ε, b) = 1
w(a, ε) = 1

Установка Elasticsearch

Prerequisites:
иметь установленную Java >= version 7

https://www.elastic.co/downloads/elasticsearch

bin/elasticsearch

http://localhost:9200/

pip install elasticsearch

Mappings

```
PUT my_index
{
  "mappings": {
    "_doc": {
      "properties": {
        "title":    { "type": "text"  },
        "name":     { "type": "text"  },
        "age":      { "type": "integer" },
        "created":  {
          "type":   "date",
          "format": "strict_date_optional_time||epoch_millis"
        }
      }
    }
  }
}
```

Создание и заполнение индекса

```
PUT http://localhost:9200/blogs

{
   "settings" : {
      "index" : {
         "number_of_shards" : 5, "number_of_replicas" : 3
      }
   }
}

POST http://localhost:9200/blogs/_bulk

{
   "index":{
      "_index":"blogs", "_type":"post", "_id":"10"
   }
}
{
   "title":"Test1", "description":"First test description"
}
```

Получение результатов

```
GET http://localhost:9200/schools/school/1

GET http://localhost:9200/index1,index2,index3/_search

{
    "query" : {
        "match" : { "title": "test" }
    }
}

GET http://localhost:9200/_search?q=name:central
```

Синтаксис запросов

```
+ signifies AND operation
| signifies OR operation
- negates a single token
" wraps a number of tokens to signify a phrase for searching
* at the end of a term signifies a prefix query
( and ) signify precedence
~N after a word signifies edit distance (fuzziness)
~N after a phrase signifies slop amount
```

Внедряем в приложение. Вариант 1

https://elasticsearch-py.readthedocs.io/en/master/

```
>>> from elasticsearch import Elasticsearch
>>> es = Elasticsearch()
>>> es.indices.create(index='my-index', ignore=400)

>>> es.index(index="my-index", id=42, body={"any": "data", "timestamp":
datetime.now()})
{'_index': 'my-index',
 '_type': '_doc',
 '_id': '42',
 '_version': 1,
 'result': 'created',
 '_shards': {'total': 2, 'successful': 1, 'failed': 0},
 '_seq_no': 0,
 '_primary_term': 1}

# but not deserialized
>>> es.get(index="my-index", id=42)['_source']
```

Внедряем в приложение. Вариант 2

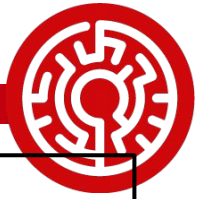
https://github.com/myarik/django-rest-elasticsearch

```
from elasticsearch_dsl import Document, Date, Integer, Keyword, Text, GeoPoint

class BlogIndex(Document):
    pk = Integer()
    title = Text(fields={'raw': Keyword()})
    created_at = Date()
    body = Text()
    tags = Keyword(multi=True)
    is_published = Boolean()
    location = GeoPoint()

    class Index:
        name = 'blog'


BlogIndex.init()
```

Внедряем в приложение. Вариант 2

```
from rest_framework_elasticsearch import es_views, es_pagination, es_filters

class BlogView(es_views.ListElasticAPIView):
    es_client = es_client
    es_model = BlogIndex
    es_pagination_class = es_pagination.ElasticLimitOffsetPagination
    es_filter_backends = (
        es_filters.ElasticFieldsFilter,
        es_filters.ElasticFieldsRangeFilter,
        es_filters.ElasticSearchFilter,
        es_filters.ElasticOrderingFilter,
        es_filters.ElasticGeoBoundingBoxFilter
    )
    es_ordering = 'created_at'
    es_filter_fields = (
        es_filters.ESFieldFilter('tag', 'tags'),
    )
    es_range_filter_fields = (
        es_filters.ESFieldFilter('created_at'),
    )
    es_search_fields = (
        'tags',
        'title',
    )

    es_geo_location_field = es_filters.ESFieldFilter('location')
    es_geo_location_field_name = 'location'
```

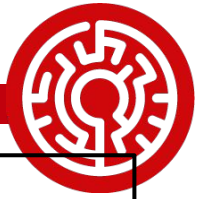
Внедряем в приложение. Вариант 3



https://github.com/sabricot/django-elasticsearch-dsl/

```
# documents.py

from django_elasticsearch_dsl import Document
from django_elasticsearch_dsl.registries import registry
from .models import Car


@registry.register_document
class CarDocument(Document):
    class Index:
        # Name of the Elasticsearch index
        name = 'cars'
        # See Elasticsearch Indices API reference for available settings
        settings = {'number_of_shards': 1,
                    'number_of_replicas': 0}

    class Django:
        model = Car # The model associated with this Document

        # The fields of the model you want to be indexed in Elasticsearch
        fields = [
            'name',
            'color',
            'description',
            'type',
        ]
```

Внедряем в приложение. Вариант 3



```
./manage.py search_index --rebuild

s = CarDocument.search().filter("term", color="blue")[:30]
qs = s.to_queryset()
```

Домашнее задание

- Написать функцию, которая будет считать расстояние Левенштейна между двумя словами
- Развернуть и наполнить тестовыми данными Elasticsearch
- Реализовать поиск по пользователям, чатам и сообщениям (прикрутить к фронтенду)

ТЕХНОТРЕК

Спасибо за внимание!