Модели стохастических объектов (методы анализа данных)

Практическая работа №1 (Генерация выборки)

КИ18-16 Прекель В.А.

Используется библиотека `FsHttp` для http-запросов и `JsonProvider` для доступа к json-данным в статически-типизированном стиле

In [1]:
```
#r "nuget: FSharp.Data, 3.3.3"
#r "nuget: SchlenkR.FsHttp, 5.0.0"

open System
open System.IO
open FSharp.Data
open FsHttp
open FsHttp.DslCE
open System.Text.Json
```

```
Installed package SchlenkR.FsHttp version 5.0.0
Installed package FSharp.Data version 3.3.3
```

JsonProvider выводит типы из вручную скачанного фрагмента

In [2]:
```
type json = JsonProvider<"sample.json">
```

Количество комментариев под 5 постом в фрагменте:

In [3]:
```
json.GetSample().Response.Items.[4].Likes.Count
```

Out[3]: 290

ServiceToken для доступа к Vk Api

In [4]:
```
let accessToken = "1bb9ca221bb9ca221bb9ca22ad1bdfa76e11bb91bb9ca22441bbfc7d2c
```

Id паблика для анализа. Положительный id - страница человека.

In [5]:
```
let ownerId = -120075923
//let ownerId = 202729931
```

Функции асинхронного получения и парсинга `count` постов со смещениев в `offset` постов

In [6]:
```
let getAsync (count: int) (offset: int) =
    httpAsync {
        GET $"https://api.vk.com/method/wall.get?v=5.78&owner_id={ownerId}&ac
    }
let getItemsAsync count offset =
    async {
        let! response = getAsync count offset
```

```
        let! content = response.content.ReadAsStreamAsync() |> Async.AwaitTas
        return json.Load(content).Response.Items
    }
```

Сколько лайков под последними 4 постами:

In [7]:
```
let y = getItemsAsync 5 0 |> Async.RunSynchronously |> Seq.map (fun i -> i.Li
y
```

Out[7]:

index	value
0	60
1	403
2	613
3	293
4	427

Структура нужных данных и функция преобразования в нужные данные

In [8]:
```
type Post =
    { OwnerId: int
      Id: int
      Date: DateTimeOffset
      AttachmentTypes: string list
      Comments: int
      Likes: int
      Reposts: int
      Views: Nullable<int>
      FromId: int
      PostSource: string
      PostType: string
      SignerId: Nullable<int>}

let responseToPost (r: json.Item) =
      { OwnerId = r.OwnerId
        Id = r.Id
        Date = DateTimeOffset.FromUnixTimeSeconds(int64 r.Date)
        AttachmentTypes =
            r.Attachments
            |> Array.map (fun t -> t.Type)
            |> List.ofArray
        Comments = r.Comments.Count
        Likes = r.Likes.Count
        Reposts = r.Reposts.Count
        Views = Option.toNullable (r.Views |> Option.map (fun v -> v.Count))
        FromId = r.FromId
        PostSource = r.PostSource.Type
        PostType = r.PostType
        SignerId = Option.toNullable r.SignerId }
```

Кол-во постов

In [9]:
```
let count =
    json
        .Load((getAsync 0 0 |> Async.RunSynchronously)
            .content.ReadAsStream())
        .Response.Count
count
```

Out[9]: 38891

Сколько требуется сделать запросов и с какими смещениями, еспри принимать по 100 постов:

In [10]:
```
let queries = (float <| count) / 100. |> ceil |> int

let offsets = [ 0 .. queries ] |> List.map ((*) 100)
let offsets_fst = [ 0 .. queries ] |> List.map ((*) 100) |> List.head |> List
(queries, offsets)
```

Out[10]:

Item1	Item2
389	[0, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, 1600, 1700, 1800, 1900 ... (370 more)]

Получение всех постов и преобразование в нужную структуру паралелльно

In [11]:
```
let items =
    offsets
    |> List.map (getItemsAsync 100) // Для всех смещений получается 100 посто
    |> Async.Parallel // Параллельно
    |> Async.RunSynchronously // Запустить это синхронно
    |> Array.concat // Склеить результат
    |> Array.map responseToPost // Преобразовать в нужную структуру
    |> Array.sortBy (fun p -> p.Date) // Отсортировтаь по дате
    |> Array.distinctBy (fun p -> p.Id) // Посты должны быть с уникальмими id
```

In [12]:
```
items
```

Out[12]:

index	OwnerId	Id	Date	AttachmentTypes	Comments	Likes	Reposts	Views	From
0	-120075923	1	2016-04-23 18:49:13Z	[photo]	28	61	1	<null>	-120075
1	-120075923	2	2016-04-23 19:18:26Z	[photo]	1	122	2	<null>	-120075
2	-120075923	3	2016-04-23 19:35:32Z	[photo]	0	225	9	<null>	-120075
3	-120075923	4	2016-04-23 19:35:39Z	[photo]	9	38	0	<null>	-120075
4	-120075923	7	2016-04-23 20:24:46Z	[ ]	9	45	0	<null>	-120075
5	-120075923	23	2016-04-24 11:57:00Z	[photo, photo, photo, photo, photo, photo, photo, photo, photo, photo]	6	57	0	<null>	-120075
6	-120075923	37	2016-04-24 13:42:00Z	[photo, photo, photo, photo, photo, photo, photo, photo, photo, photo]	2	395	22	<null>	-120075

index	OwnerId	Id	Date	AttachmentTypes	Comments	Likes	Reposts	Views	Fror
7	-120075923	40	2016-04-24 15:22:23Z	[photo, photo]	7	23	0	<null>	-120075
8	-120075923	52	2016-04-24 17:32:04Z	[photo]	4	41	2	<null>	-120075
9	-120075923	61	2016-04-25 05:30:37Z	[photo]	3	35	0	<null>	-120075
10	-120075923	64	2016-04-25 05:40:37Z	[photo]	5	36	0	<null>	-120075
11	-120075923	69	2016-04-25 08:16:09Z	[photo]	1	19	0	<null>	-120075
12	-120075923	71	2016-04-25 08:19:06Z	[photo]	6	18	0	<null>	-120075
13	-120075923	79	2016-04-25 09:14:01Z	[photo, photo]	9	40	1	<null>	-120075
14	-120075923	83	2016-04-25 09:35:00Z	[photo, photo]	2	141	4	<null>	-120075
15	-120075923	89	2016-04-25 10:15:05Z	[photo]	4	29	0	<null>	-120075
16	-120075923	92	2016-04-25 10:40:26Z	[photo]	7	25	0	<null>	-120075
17	-120075923	98	2016-04-25 11:15:12Z	[photo]	1	22	0	<null>	-120075
18	-120075923	103	2016-04-25 12:27:12Z	[photo]	2	31	0	<null>	-120075
19	-120075923	105	2016-04-25 13:01:00Z	[photo, link]	1	49	0	<null>	-120075

Сохранение в файл

In [13]:

```
let out = new StreamWriter "memeblog.json"
fprintfn out "%s" (JsonSerializer.Serialize <| items)
out.Close()
```