Федеральное государственное автономное образовательное учреждение
высшего профессионального образования

**«Национальный исследовательский университет
«Высшая школа экономики»**

Кафедра Анализа данных и искусственного интеллекта

Отчёт о прохождении учебной/производственной практики

Студент: Киреев Руслан Юрьевич

Группа: 371ПМИ

Руководитель Яворский Ростислав Эдуардович

Москва, 2013

1. Оглавление

2. Постановка задачи

Составить граф, где вершины – пользователи форума http://forum.hse.ru/newforum/index.php?t=thread&frm_id=125&S=fc9b23190ad581050d7c22397d287ea7, а ребра – комментирование одного пользователя другим. Визуализировать его с помощью пакета Gephi

3. Практическое значение задачи

Визуализация данных позволяет наглядно показать активность каждого участника форума.

4. Ход решения задачи

Одно из возможных решений задачи – используя какой-нибудь язык программирования, написать программу, которая проходит по всем темам форума, в каждой из которых получает список пользователей, которые оставили сообщения, и дерево, кто кого комментировал, заносит все эти данные в отдельный файл в виде списка ребер.

В качестве используемого языка я взял Python. Используя библиотеку BeautifulSoup, я, для каждой из четырех страниц форума, получал html код страницы, где находил все теги <a> класса 'big', откуда извлекал ссылки на темы. Каждую тему из полученного списка я открывал в режиме просмотра сообщений в виде дерева. Там я находил все теги <td> класса 'Gentext nw wa vt', получал атрибут 'padding-left: ', чье значение, кратное 15, дает нам информацию, какого пользователя комментирует текущий пользователь (если текущее значение, например, 30, а предыдущее 15, то текущий пользователь отвечает предыдущему, а если, например, у текущего – 15, а у предыдущего – 30, то текущий отвечает первому (если идти вверх по списку сообщений) пользователю со значением 0), также я получаю имена пользователей, которые содержатся во втором <a> в теге <td>.

Таким образом, я имею список всех пользователей, отписавшихся на форуме, и список ребер, соответствующих, кто кому отвечал. Чтобы было удобно находить на графе определенное имя, я присваиваю каждому определенный цвет в соответствии с его лексикографическим порядком.

Полученные данные я записываю в файл с расширением .gdf по правилам, описанным здесь http://gephi.org/users/supported-graph-formats/gdf-format/

Полученный файл, я отрываю с помощью Gephi, импортирую данные в рабочую область программы, здесь я настраиваю отображение ребер кривыми, имена вершин, их размер (зависимый от мощности вершины), применяю алгоритм Force Atlas для укладки

графа, чтобы он лучше читался. Результат сохраняю в виде векторного изображения формата .svg.

5. Заключение

В ходе выполнения практики я приобрел навыки работы с языком Python, библиотекой BeautifulSoup и средой визуализации Gephi.

6. Приложение

Приложение 1. Код скрипта, который получает граф комментирования с форума, написанный на языке Python.

```
from BeautifulSoup import BeautifulSoup
import urllib


list_of_edges=[]
list_of_nodes=[]
url = 'http://forum.hse.ru/newforum/index.php?
t=thread&frm_id=125&S=113870092c010050c608f900a805de56&start='
page_number = ['0', '40', '80', '120']
for i in range(4):
    page = urllib.urlopen(url + page_number[i])
    soup = BeautifulSoup(page.read())
    for each_a in soup.findAll('a', {'class':'big'}):
        thread = each_a.get('href')
        thread = "http://forum.hse.ru/newforum/" + thread
        thread = thread.replace('msg', 'tree')
        page = urllib.urlopen(thread)
        soup = BeautifulSoup(page.read(), fromEncoding="cp1251")
        users = []
        previous_padding = -1;
        for each_td in soup.findAll('td', {'class':'Gentext nw wa vt'}):
            current_padding = each_td['style']
            current_padding = current_padding.replace('px', '');
            current_padding = current_padding.replace('padding-left: ', '');
            current_padding = int(current_padding)
            current_user = str(each_td.findAll('a')[1].string)
            if current_user not in list_of_nodes:
                list_of_nodes.append(current_user)
            flag = 1
            while current_padding <= previous_padding:
                flag = 0
                user1 = users[-1]
```

```
                users.pop(-1)
                user2 = users[-1]
                temp_list = []
                temp_list.append(user1)
                temp_list.append(user2)
                list_of_edges.append(temp_list)
                previous_padding = previous_padding - 15
            if flag:
                previous_padding = current_padding
            users.append(current_user)
        while len(users) > 1:
            user1 = users[-1]
            users.pop(-1)
            user2 = users[-1]
            temp_list = []
            temp_list.append(user1)
            temp_list.append(user2)
            list_of_edges.append(temp_list)

output = open('graph_replies.gdf', 'w')
output.write('nodedef>name VARCHAR,label VARCHAR,color VARCHAR' + '\n')
list_of_nodes.sort()
r = 0;
g = 0;
b = 100;
for each_node in list_of_nodes:
    rgb = str(r) + ',' + str(g) + ',' + str(b)
    r = r + 4;
    if r >= 255:
        r = 255
        g = g + 4
    if g >= 255:
        g = 255
        r = r - 8
    output.write(each_node + ',')
    output.write(each_node + ',')
    output.write("'" + rgb + "'\n")
output.write('edgedef>node1 VARCHAR,node2 VARCHAR' + '\n')
for each_edge in list_of_edges:
    output.write(each_edge[0] + ',')
```

```
    output.write(each_edge[1] + '\n')
output.close()
```