# ДОДАТОК А
(довідковий)
# ПРОГРАМНИЙ КОД

## А1 – отримання додаткової інформації та побудова графіків.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;
using TweetSharp;

namespace Avrora
{
    public partial class Wind1 : Form
    {

        private UserStat userStat;
        public Series graf = new Series();
        public Series Min = new Series();
        public Series Max = new Series();
        List<double> marks = new List<double>();
        List<DateTime> datas = new List<DateTime>();
        List<int> maxs = new List<int>();
        List<int> mins = new List<int>();
        public Wind1(User _user)
        {
            InitializeComponent();
            bool byaka = false;
            foreach (var a in UserStatList.Stats)
            {
                if (a.name.Equals(_user.FullName))
                {
                    label5.Text = _user.FullName;
                    label6.Text = _user.Birthday;
                    label7.Text = _user.TwitterID.ToString();
                    label8.Text = _user.TwitterNick;
                    label12.Text = a.avg.ToString();
                    label10.Text = a.diff.ToString();
                    label18.Text = a.period.ToString();
                    label14.Text = a.negToAll.ToString();
                    label16.Text = a.posToAll.ToString();
                    var sentiment = Sentiment.Instance;
                    var twits =
TweCon.service.ListTweetsOnHomeTimeline(new
```

```
ListTweetsOnHomeTimelineOptions { ExcludeReplies = true, Count =
1000 });
                    foreach (var tweet in twits)
                    {
                        if
(tweet.User.ScreenName.Equals(_user.TwitterNick))
                            if
(!tweet.Text.ToString().StartsWith("RT"))
                            {

listBox1.Items.Add(tweet.CreatedDate.ToString() + " " +
tweet.Text.ToString());
                                var score =
sentiment.GetScore(tweet.Text.ToString());

//listBox2.Items.Add(score.Avg.ToString());
                                marks.Add(score.Avg);
                                datas.Add(tweet.CreatedDate);

                                maxs.Add(score.Max);

                                mins.Add(score.Min);

                            }
                    }
                    Series mySeriesOfPoint = new Series("Graf");
                    Series min = new Series("Min");
                    Series max = new Series("Max");
                    min.ChartType = SeriesChartType.Line;
                    max.ChartType = SeriesChartType.Line;
                    mySeriesOfPoint.ChartType =
SeriesChartType.Line;
                    mySeriesOfPoint.ChartArea = "ChartArea1";
                    for (int i = 0; i < marks.Count(); i++)
                    {
                        mySeriesOfPoint.Points.AddXY(datas[i],
marks[i]);
                        min.Points.AddXY(datas[i], mins[i]);
                        max.Points.AddXY(datas[i], maxs[i]);
                    }
                    chart1.Series[0] = mySeriesOfPoint;
                    //chart1.Series.Add(min);
                    //chart1.Series.Add(max);
                    graf = mySeriesOfPoint;
                    Min = min;
                    Max = max;
                    byaka = true;
                }
            }
            if (byaka == false)
            {
                userStat = new UserStat(_user.FullName);
                label5.Text = _user.FullName;
                label6.Text = _user.Birthday;
                label7.Text = _user.TwitterID.ToString();
                label8.Text = _user.TwitterNick;
```

```
                var sentiment = Sentiment.Instance;
                var twits =
TweCon.service.ListTweetsOnHomeTimeline(new
ListTweetsOnHomeTimelineOptions { ExcludeReplies = true, Count =
1000 });
                foreach (var tweet in twits)
                {
                    if
(tweet.User.ScreenName.Equals(_user.TwitterNick))
                        if (!tweet.Text.ToString().StartsWith("RT"))
                        {

listBox1.Items.Add(tweet.CreatedDate.ToString() + " " +
tweet.Text.ToString());
                            var score =
sentiment.GetScore(tweet.Text.ToString());

//listBox2.Items.Add(score.Avg.ToString());
                            marks.Add(score.Avg);
                            datas.Add(tweet.CreatedDate);

                            maxs.Add(score.Max);

                            mins.Add(score.Min);

                        }
                }
                Series mySeriesOfPoint = new Series("Graf");
                Series min = new Series("Min");
                Series max = new Series("Max");
                min.ChartType = SeriesChartType.Line;
                max.ChartType = SeriesChartType.Line;
                mySeriesOfPoint.ChartType = SeriesChartType.Line;
                mySeriesOfPoint.ChartArea = "ChartArea1";
                for (int i = 0; i < marks.Count(); i++)
                {
                    mySeriesOfPoint.Points.AddXY(datas[i],
marks[i]);
                    min.Points.AddXY(datas[i], mins[i]);
                    max.Points.AddXY(datas[i], maxs[i]);
                }
                chart1.Series[0] = mySeriesOfPoint;
                //chart1.Series.Add(min);
                //chart1.Series.Add(max);
                graf = mySeriesOfPoint;
                Min = min;
                Max = max;
                double minVal = 7;
                double maxVal = -7;
                int countOfNeg = 0;
                double count = 0;
                foreach (double a in marks)
                {
                    count += a;
                    if (minVal > a)
                        minVal = a;
```

```
                    if (maxVal < a)
                        maxVal = a;
                    if (a < 0)
                        countOfNeg++;
                }
                userStat.avg = count / marks.Count;
                userStat.diff = maxVal - minVal;
                userStat.negToAll = Convert.ToDouble(countOfNeg) /
Convert.ToDouble(marks.Count);
                userStat.posToAll = 1 - userStat.negToAll;

                DateTime prev = new DateTime();
                bool first = true;
                int dayS = 0;
                foreach (DateTime a in datas)
                {
                    if (first)
                    {
                        prev = a.Date;
                        dayS++;
                        first = false;
                    }
                    else
                        if (!a.Date.Equals(prev))
                    {
                        prev = a.Date;
                        dayS++;
                    }
                }
                userStat.period = Convert.ToDouble(datas.Count) /
Convert.ToDouble(dayS);
                label12.Text = userStat.avg.ToString();
                label10.Text = userStat.diff.ToString();
                label18.Text = userStat.period.ToString();
                label14.Text = userStat.negToAll.ToString();
                label16.Text = userStat.posToAll.ToString();

                foreach (var s in UserStatList.Stats)
                    if (s.name.Equals(_user.FullName))
                    {

                        UserStatList.Stats.Remove(s);
                    }

                UserStatList.Stats.Add(userStat);

            }
        }

        private void CloseBut_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
```

```
        {

        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (radioButton1.Checked)
            {
                Graf a = new Graf(graf, Min, Max);

                a.ShowDialog();
            }
            else
            {
                Graf a = new Graf(graf);

                a.ShowDialog();
            }
        }

        private void radioButton1_CheckedChanged(object sender,
EventArgs e)
        {
            if (radioButton1.Checked == true)
            {
                chart1.Series.Add(Min);
                chart1.Series.Add(Max);
            }
            else
            {
                chart1.Series.Clear();
                chart1.Series.Add(graf);
            }
        }

        private void label1_Click(object sender, EventArgs e)
        {
            radioButton1.Checked = false;
        }


    }
}
```

## А2 – клас для роботи з AFINN.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text.RegularExpressions;
using System.IO;

namespace Avrora
{
```

```
    class Sentiment
    {
        #region Fields
         private static volatile Sentiment _instance;
         private static readonly object SyncRoot = new Object();
         private readonly IDictionary<string, int> _words;
        #endregion

        #region Properties
         public IDictionary<string, int> Words { get { return
_words; } }
         public int WordsCount { get { return _words.Count; } }
        #endregion
        public static Sentiment Instance
         {

             get
             {
                 if (_instance == null)
                 {
                     lock (SyncRoot)
                     {
                         if (_instance == null)
                             _instance = new Sentiment();
                     }
                 }


                 return _instance;
             }
         }

        #region ctor
         private Sentiment()
         {
             _words = new Dictionary<string, int>();


             using (var file = new StreamReader("AFINN-111.txt"))
             {
                 string line;
                 while ((line = file.ReadLine()) != null)
                 {
                     var bits = line.Split('\t');
                     _words.Add(bits[0], int.Parse(bits[1]));
                 }
             }
         }
        #endregion

        private static IEnumerable<string> Tokenize(string input)
         {
             input = Regex.Replace(input, "[^a-zA-Z ]+", "");
            input = Regex.Replace(input, "0123456789+", "");
             input = Regex.Replace(input, @"\s+", " ");
```

```
            input = input.ToLower();
            return input.Split(' ');
        }
        public Score GetScore(string input)
        {
            var score = new Score { Tokens = Tokenize(input) };

            double avg = 0;
            int min = 0;
            int max = 0;
            bool first = true;
            foreach (var token in score.Tokens)
            {
                if (!_words.ContainsKey(token)) continue;


                var item = _words[token];
                score.Words.Add(token);
                if(first)
                {
                    avg = item;
                    first = false;
                    min = item;
                    max = item;
                }
                else
                {
                    if (item > max)
                        max = item;
                    if (item < min)
                        min = item;

                    avg = (avg + Convert.ToDouble(item)) / 2;
                }
                if (item > 0) score.Positive.Add(token);
                if (item< 0) score.Negative.Add(token);

                score.Sentiment += item;
            }
            score.Avg = avg;
            score.Min = min;
            score.Max = max;
            return score;
        }
        public void InjectWords(IDictionary<string, int> words)
        {
            foreach (var word in words.Where(word
=> !_words.ContainsKey(word.Key)))
            {
                _words.Add(word.Key, word.Value);
            }
        }

        #region Inner Score Class
        public class Score
```

```
        {
            /// <summary>
            /// Tokens which were scored
            /// </summary>
            public IEnumerable<string> Tokens { get; set; }
           public double Avg { get; set; }
           public int Min { get; set; }
           public int Max { get; set; }
            /// <summary>
            /// Total sentiment score of the tokens
            /// </summary>
            public int Sentiment { get; set; }
           /// <summary>
           /// Average sentiment score Sentiment/Tokens.Count
           /// </summary>
           public double AverageSentimentTokens { get { return
(double)Sentiment / Tokens.Count(); } }
            /// <summary>
            /// Average sentiment score Sentiment/Words.Count
            /// </summary>
            public double AverageSentimentWords { get { return
(double)Sentiment / Words.Count(); } }
            /// <summary>
            /// Words that were used from AFINN
            /// </summary>
            public IList<string> Words { get; set; }
            /// <summary>
            /// Words that had negative sentiment
            /// </summary>
            public IList<string> Negative { get; set; }
            /// <summary>
            /// Words that had positive sentiment
            /// </summary>
            public IList<string> Positive { get; set; }


            public Score()
            {
                Words = new List<string>();
                Negative = new List<string>();
                Positive = new List<string>();
            }
        }
        #endregion




    }
}
```

А3 – пошук користувача в мережі Twitter та реалізація функції “підписатися на оновлення”.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TweetSharp;

namespace Avrora
{
    public partial class TwiConnection : Form
    {
        private List<long> ids = new List<long>();
        private SearchForUserOptions options = new
SearchForUserOptions { Q = TweCon.usName};
        public TwiConnection()
        {
            InitializeComponent();
            label2.Text = TweCon.usName;
        }

        private void label1_Click(object sender, EventArgs e)
        {

var SearchRes = TweCon.service.SearchForUser(options);
            long curId;
            if (SearchRes == null)
            {
                Error a = new Error();
                a.ShowDialog();
                //Close();
            }
            else
            {
                foreach (var user in SearchRes)
                {

listOfUsers.Items.Add(user.ScreenName.ToString());
                    curId = user.Id;
                    ids.Add(curId);
                }
                progressBar1.Visible = false;
            }

            button1.Visible = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }
```

```
        private void followButton_Click(object sender, EventArgs e)
        {
            String UserName = listOfUsers.SelectedItem.ToString();
            TweCon.service.BeginFollowUser(new FollowUserOptions
{ ScreenName = UserName });
            TweCon.usName = UserName;
            TweCon.usId = ids[listOfUsers.SelectedIndex];
            TweCon.IsSuccess = true;
            Close();
        }


    }
}
```

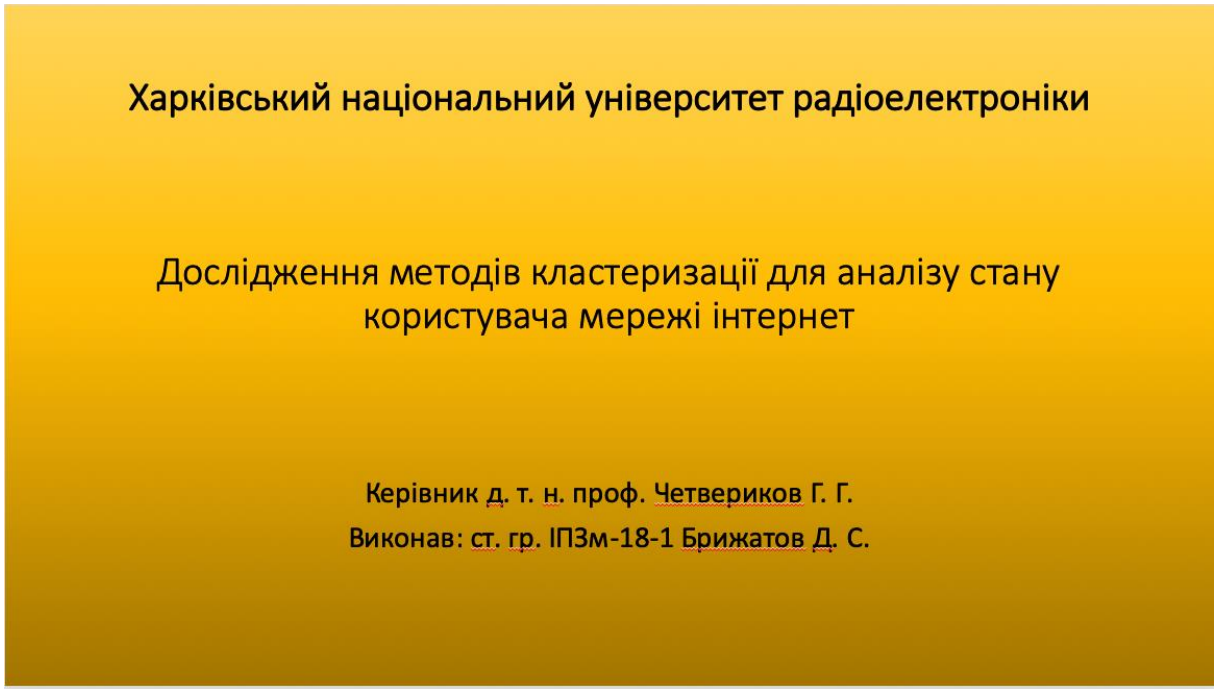
А4 – функціонал головного вікна програми, в тому числі – збереження даних в файлі.

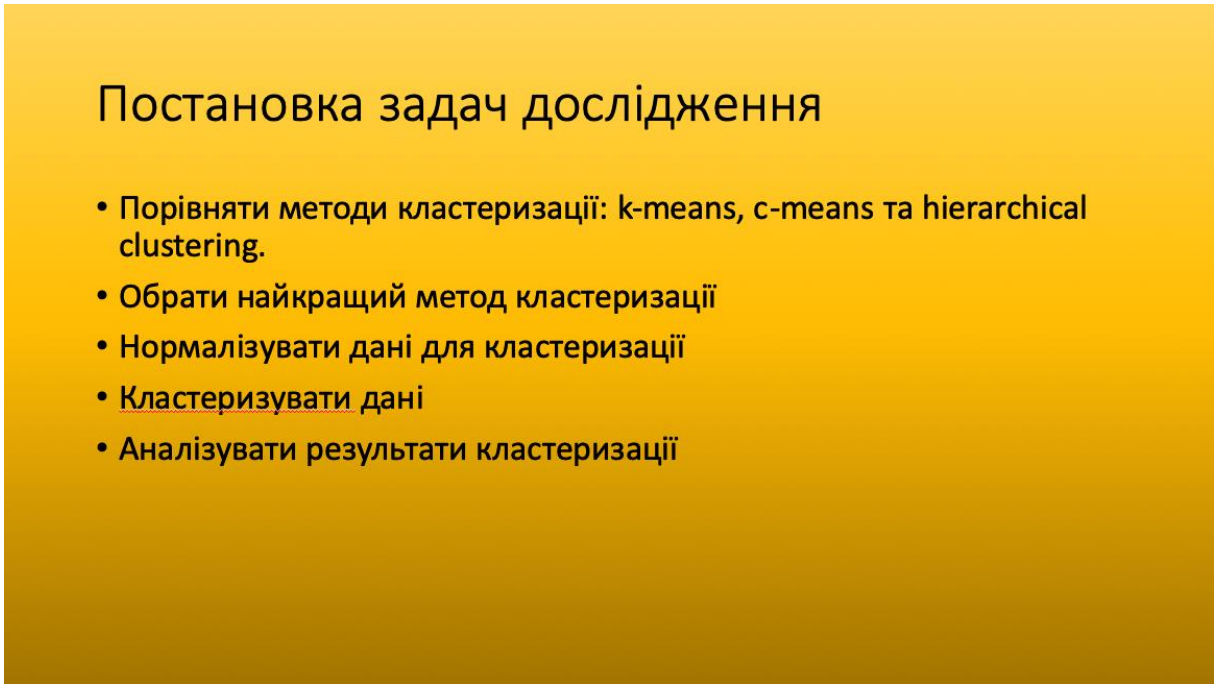
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using TweetSharp;
using Excel = Microsoft.Office.Interop.Excel;

namespace Avrora
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            //var followedList = TweCon.service.ListFriends(new
ListFriendsOptions { });
            UserList.Users.Add(new User(0, "Natkovich",
"21.03.1996"));
            UserStatList.Stats.Add(new UserStat("Natkovich", 4.7,
0.95,0.05 , 6, 0.5));

            foreach (User user in UserList.Users)
                {
                listBox1.Items.Add(user.ToString());
                }
            foreach (UserStat s in UserStatList.Stats)
            {

                dataGridView1.Rows.Add(s.name, s.avg, s.posToAll,
s.negToAll, s.diff, s.period);
            }
```

```
            listBox1.SelectedIndex = 0;
            //string url1 =
OAuthWeb.GetAuthorizationUrl("Facebook");
            //string url2 = OAuthWeb.GetAuthorizationUrl("Twitter");
        }
        private void button1_Click(object sender, EventArgs e)
        {
            AddNew a = new AddNew();
            a.ShowDialog();
            listBox1.Items.Clear();
            foreach (User user in UserList.Users)
            {
                listBox1.Items.Add(user.ToString());

            }

        }
        private void button3_Click(object sender, EventArgs e)
        {
            Wind1 b = new
Wind1(UserList.Users[listBox1.SelectedIndex]);
            b.ShowDialog();
            dataGridView1.Rows.Clear();
            foreach (UserStat s in UserStatList.Stats)
            {
                dataGridView1.Rows.Add(s.name, s.avg, s.posToAll,
s.negToAll, s.diff, s.period);
            }
        }

        private void SaveBtn_Click(object sender, EventArgs e)
        {
            SaveFileDialog a = new SaveFileDialog();
            a.Filter = "All files (*.*)|*.*|Microsoft Office Excel
*.xlsx|*.xlsx";
            if (a.ShowDialog() == DialogResult.OK)
            {
                string path = a.FileName;
                Excel.Application app = new Excel.Application();
                app.Visible = false;
                Excel.Workbook wb = app.Workbooks.Add(1);
                Excel.Worksheet ws =
(Excel.Worksheet)wb.Worksheets[1];
                // changing the name of active sheet
                ws.Name = "Exported from gridview";

                ws.Rows.HorizontalAlignment =
HorizontalAlignment.Center;
                // storing header part in Excel
                for (int i = 1; i < dataGridView1.Columns.Count + 1;
i++)
                {
                    ws.Cells[1, i] = dataGridView1.Columns[i -
1].HeaderText;
                }
```

```
                // storing Each row and column value to excel sheet
                for (int i = 0; i < dataGridView1.Rows.Count - 1;
i++)
                {
                    for (int j = 0; j < dataGridView1.Columns.Count;
j++)
                    {
                        ws.Cells[i + 2, j + 1] =
dataGridView1.Rows[i].Cells[j].Value.ToString();
                    }
                }

                // sizing the columns
                ws.Cells.EntireColumn.AutoFit();

                // save the application
                wb.SaveAs(path, Type.Missing, Type.Missing,
Type.Missing, Type.Missing, Type.Missing,
Microsoft.Office.Interop.Excel.XlSaveAsAccessMode.xlExclusive,
Type.Missing, Type.Missing, Type.Missing, Type.Missing);

                // Exit from the application
                app.Quit();
            }
        }
    }
}
```

ДОДАТОК Б
(довідковий)
**СЛАЙДИ ПРЕЗЕНТАЦІЇ**



## Результати методу попарних порівнянь

| Критерії / Альтернативи | Точ. | Склад. | Необ. кіл. | Універ. | Роз. Роз. | Вектор пріоритетів |
|---|---|---|---|---|---|---|
| c-means | 0,276 | 0,276 | 0,4 | 0,211 | 0,322 | 0,283 |
| k-means | 0,128 | 0,595 | 0,4 | 0,084 | 0,146 | 0,2 |
| Hierarchical clustering | 0,595 | 0,128 | 0,2 | 0,704 | 0,532 | 0,76 |

## Метод k-means

- Ітераційний метод кластеризації.

- Переваги:
  1. Точність, але є одне “но”.

- Недоліки:
  1. Потребує знати кількість кластерів
  2. Потребує знати кількість ітерацій
  3. При кожному запуску може давати різні результати, “но”…

# Метод c-means

- Визначає ймовірність належність вектору до кластеру.

- Переваги:

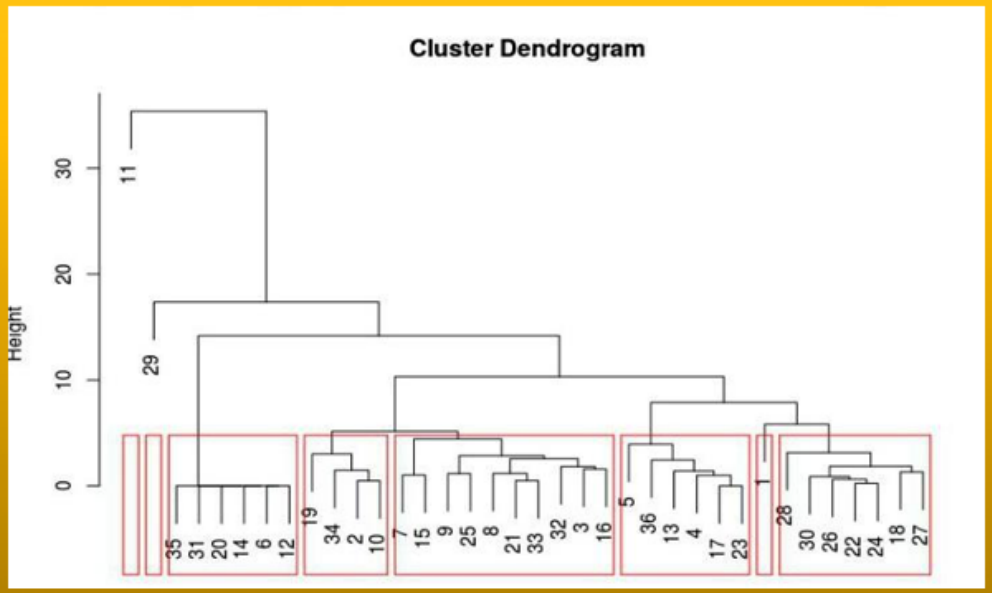
1. Вирішує проблему k-means

- Недоліки:

1. Неточність
2. Вимагає велику кількість обчислювальних ресурсів
3. Вимагає знати кількість кластерів.

# Метод hierarchical clustering

- Послідовно об'єднує менші кластери у більші



## Чому hierarchical clustering підходить?

- Не потребує знати кількість кластерів
- Точний
- Наглядний (може бути недоліком)
- Відносно швидкий, $O(n^2)$

## Нормалізація даних

- За допомогою бібліотеки семантичного аналізу (AFINN) виконується оцифровка текстових даних.
- Приклад:

1. fool (дурень) оцінюється в -2
2. clever (розумний) оцінюється в 2

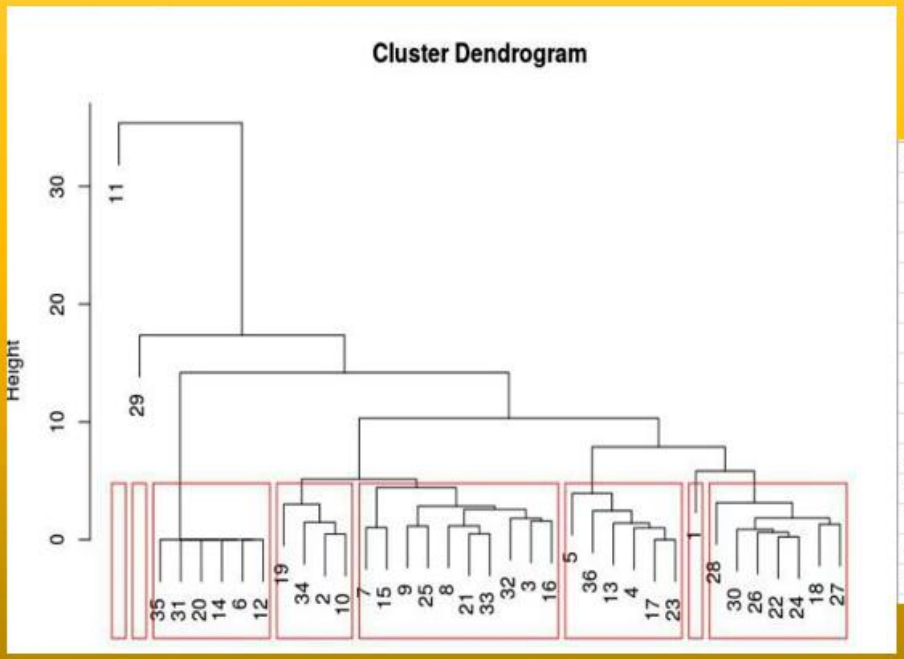
# Ознаки для кластеризації

- Середня емоційне навантаження серед всіх його повідомлень (Common Average).
- Максимальна і мінімальна різниця між першим пунктом (Difference).
- Середня кількість повідомлень у день (Periodicity).

# Результати роботи програми

| № | Full Name | Common Average | Positive % | Negative % | Difference | Periodicity |
|---|---|---|---|---|---|---|
| 1 | Bryzhatov | 0.57 | 0.79 | 0.21 | 4 | 2.3 |
| 2 | Alice | 0.4964 | 0.845 | 0.1549 | 8 | 7.1 |
| 3 | Anton | -0.2941 | 0.7647 | 0.2352 | 5.5 | 4.25 |
| 4 | Marry | 1 | 1 | 0 | 0 | 1 |
| 5 | Kirill | 2.5 | 1 | 0 | 1 | 1.5 |
| 6 | Elizabet | NaN | NaN | NaN | NaN | NaN |
| 7 | Dmitry | 0.5438 | 0.7926 | 0.2073 | 9 | 4.68 |
| 8 | Dmitry | 0.8151 | 0.983 | 0.02 | 7 | 2.5 |
| 9 | Boris | 0.7096 | 0.8115 | 0.1846 | 7 | 4.64 |
| 10 | Illya | 0.9731 | 0.8309 | 0.169 | 9 | 10.1 |
| 11 | Alex | 0.2043 | 0.798 | 0.2019 | 7 | 37 |
| 12 | Vadim | NaN | NaN | NaN | NaN | NaN |
| 13 | Igor | 0.3 | 1 | 0 | 0 | 2 |
| 14 | Daria | NaN | NaN | NaN | NaN | NaN |

# Аналіз результатів кластеризації




| № | Full Name | Common Average | Positive % | Negative % | Difference | Periodicity |
|---|---|---|---|---|---|---|
| 1 | Bryzhatov | 0.57 | 0.79 | 0.21 | 4 | 2.3 |
| 2 | Alice | 0.4964 | 0.845 | 0.1549 | 8 | 7.1 |
| 3 | Anton | -0.2941 | 0.7647 | 0.2352 | 5.5 | 4.25 |
| 4 | Marry | 1 | 1 | 0 | 0 | 1 |
| 5 | Kirill | 2.5 | 1 | 0 | 1 | 1.5 |
| 6 | Elizabet | NaN | NaN | NaN | NaN | NaN |
| 7 | Dmitry | 0.5438 | 0.7926 | 0.2073 | 9 | 4.68 |
| 8 | Dmitry | 0.8151 | 0.983 | 0.02 | 7 | 2.5 |
| 9 | Boris | 0.7096 | 0.8115 | 0.1846 | 7 | 4.64 |
| 10 | Illya | 0.9731 | 0.8309 | 0.169 | 9 | 10.1 |
| 11 | Alex | 0.2043 | 0.798 | 0.2019 | 7 | 37 |
| 12 | Vadim | NaN | NaN | NaN | NaN | NaN |
| 13 | Igor | 0.3 | 1 | 0 | 0 | 2 |
| 14 | Daria | NaN | NaN | NaN | NaN | NaN |

# Висновки

- Не зрозуміло як обробляти випадки сааркazма / жартів або провокацій (які робляться для залучення уваги, блеф).
- Психологічний стан користувача можна кластеризувати.
- Ознаки кластеризації підібрані вірно.
- Для повної автоматизації семантичного аналізу потрібно написати нейронну мережу.

ДОДАТОК В
(довідковий)
**ВІДГУК ТА РЕЦЕНЗІЇ**

**Рецензія**

на атестаційну роботу магістра
студента групи ІПЗм-18-1 Брижатова Дмитра Сергійовича
спеціальність – 121– Інженерія програмного забезпечення
освітньо-наукова програма «Інженерія програмного забезпечення»
«Дослідження методів кластеризації для аналізу стану користувача мережі інтернет»
(Тема атестаційної роботи)

Структура атестаційної роботи: пояснювальна записка 62 сторінки; програмне застосування (прикладна програма) 204 файлів загальним обсягом 76.6 Мбайт.

Представлений прототип системи для аналізу психологічного стану користувачів мережі Інтернет відповідає вимогам якості програмного забезпечення. Актуальність роботи зумовлена необхідністю створення програмного застосунку для запобігання терористичних актів.

Пояснювальна записка відповідає стандартам та вимогам, які висуваються для атестаційних робіт магістрів. Розділи пояснювальної записки пропорційні, містять матеріал у відповідності до теми. У роботі автор продемонстрував вміння працювати з науковою літературою, продемонстрував послідовний аналіз та обґрунтування вибору методу кластеризації.

Прийняті рішення для середовища для розробки програмного застосунку є вірними. Технології розробки, метод кластеризації та засіб візуалізації результатів кластеризації є обґрунтованими та зваженими.

До недоліків слід віднести не зовсім зручне використання програмного застосунку, тобто потрібно більше автоматизації процесу збору та кластеризації семантичних даних користувача. Також для подальшого розвитку роботи потрібно створити нейрону мережу, так як представлений аналіз кластеризації є прикладом аналізу даних людиною. Це не є принципово важливим недоліком, але така мережа є наступним кроком у розвитку роботи.

В цілому атестаційна робота магістранта групи ІПЗм-18-1 Брижатова Д. С. відповідає вимогам до атестаційних робіт і заслуговує оцінки «добре» (80 балів, С). Атестаційну роботу можна представити для захисту в ЕК за спеціальністю 121 – Інженерія програмного забезпечення, освітньо-науковою програмою «Інженерія програмного забезпечення».

Рецензент
професор кафедри інформатики,
д.т.н. проф. Гороховатський В. О.

**Рецензія**

на атестаційну роботу магістра
студента групи ІПЗм-18-1 Брижатова Дмитра Сергійовича
спеціальність – 121– Інженерія програмного забезпечення
освітньо-наукова програма «Інженерія програмного забезпечення»
«Дослідження методів кластеризації для аналізу стану користувача мережі інтернет»

(Тема атестаційної роботи)

Структура атестаційної роботи: пояснювальна записка 62 сторінки; програмне застосування (прикладна програма) 204 файлів загальним обсягом 76.6 Мбайт.

В атестаційній роботі представлено прототип системи для аналізу психологічного стану користувачів мережі Інтернет. Семантичний аналіз користувачів виконувався за допомогою семантичного словника AFFIN. Розроблена програмна система відповідає такими основними атрибутам якості програмного забезпечення, як продуктивність, доступність, надійність та безпека. На сьогодні, в умовах терористичних загроза та збільшення психічних проблем людей, питання пошуку потенційно небезпечних для соціуму організацій та людей є достатньо актуальною.

Пояснювальна записка до розробленого проекту відповідає усім вимогам, що висуваються для атестаційних робіт: правильно оформлена згідно до госту, доцільно розміщені текст та графічні матеріали.

Аналіз предметної галузі, атрибутів якості програмного забезпечення, архітектурних стилів і технології було поведено на високому рівні. Варто відзначити доцільність обраних студентом методів і засобів розробки, тестування і впровадження програмного забезпечення. Усі прийняті технічні рішення є обґрунтованими і зваженими.

Результати дослідження кластеризації психологічного стану користувача та програмний застосунок можуть бути використані прі створенні системи зі схожими функціональними і нефункціональними вимогами.

Серед недоліків атестаційної роботи магістра варто відзначити розбиття програмного застосунку на дві частини. Перша частина для семантичного аналізу, друга для кластеризації. Програмний застосунок акумулює всі дані користувача за весь період, але було б краще розбити аналіз на фіксовані незалежні інтервали. Проте зазначені недоліки значно не впливають на якість представленої роботи.

В цілому атестаційна робота магістранта групи ІПЗм-18-1 Брижатова Д. С. відповідає вимогам до атестаційних робіт і заслуговує «добре -75». Атестаційну роботу можна представити для захисту в ЕК за спеціальність 121 – Інженерія програмного забезпечення, освітньо-науковою програмою «Інженерія програмного забезпечення».

Рецензент
Зав кафедри інформаційних технологій проектування
Національного аерокосмічного університету ім. М.Є. Жуковського "ХАІ",
д.т.н., проф. Дружинін Є. А.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет комп'ютерних наук

# ВІДГУК

на атестаційну роботу магістра
Брижатова Дмитра Сергійовича,
спеціальність 121- Інженерія програмного забезпечення
освітньо-наукова програма «Інженерія програмного забезпечення»
Тема атестаційної роботи: **"Дослідження методів кластеризації для аналізу стану користувача мережі інтернет"**

Атестаційна робота магістра Брижатова Д.С. присвячена дослідженню методів кластеризації для семантичного аналізу психологічного стану користувача. За допомогою ефективний та швидкій обробки великих семантичних мета даних можливо швидко реагувати на провокативні дії різноманітних організацій та ліквідувати соціально небезпечних для соціуму людей. Семантичний аналіз здійснюється за допомогою семантичного словника AFINN. Користувачі аналізуються із соціальної мережі Twitter. У процесі роботи першої частини програми запроваджується семантичний аналіз цих користувачів, збирається різноманітна статистика та зберігається у exel файл. Згідно до цього файлу буде виконуватись кластеризація у другій частині програми. Вибір методу кластеризації є обгрунтований та доцільний, використовується метод ієрархічної кластеризації що є самим оптимальним серед розглянутих у роботі для вирішення поставлених задач. Атестаційна робота є актуальною, програмне забезпечення немає відомих, відкритих (OpenSource) аналогів.

Робота виконана якісно, самостійно. Під час розробки студент показав знання та вміння використовувати в практичній діяльності засоби розробки Visual Studio та RStudion. У процесі роботи студент продемонстрував серйозне відношення до роботи, високий рівень підготовленості до самостійних наукових досліджень та самостійної роботи, показав уміння користуватися науково-технічною літературою, ресурсами мережі Інтернет, виявив глибокі знання в області кластеризації та мов програмування C# та R.

Під час виконання атестаційної роботи була проаналізована предметна область, виявлені основні причини написання роботи, за допомогою методу попарних порівнянь виявлені пріоритетні властивості методів кластеризації, а згідно запровадженого аналізу цих методів був обраний найоптимальніший для вирішення проблеми швидкого аналізу великих обсягів семантичних даних. Студент реалізував семантичний аналіз і кластеризацію, запровадив аналіз результатів кластеризації у рамках дослідження методів кластеризації. Результати роботи відповідають завданню на атестаційну роботу і є автентичними, про що свідчить дуже низький відсоток схожості з іншими роботами в Інтернеті.

Магістрант гр. ІПЗм-18-1 Брижатов Д.С. готовий до самостійної інженерної діяльності. Атестаційну роботу можна подати до захисту в ЕК за спеціальністю 121 – «Інженерія програмного забезпечення», освітньо-науковою програмою «Інженерія програмного забезпечення».

«_____» _____ 20__р.

_____
підпис

Керівник атестаційної роботи магістра
д. т. н. проф. Четвериков Г. Г.