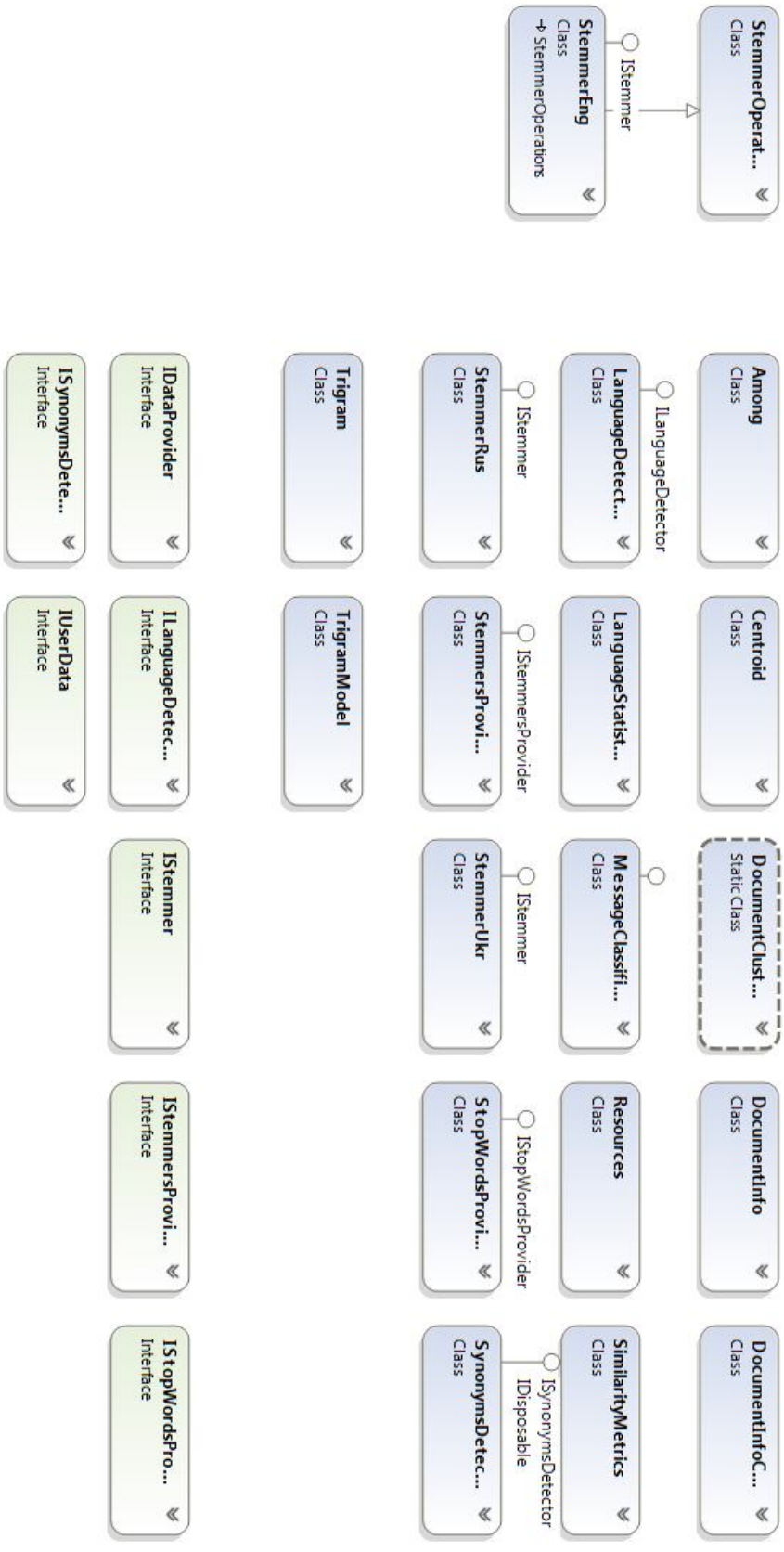
ДОДАТОК А

ДІАГРАМА КЛАСІВ МОДУЛЮ КЛАСТЕРИЗАЦІЇ



ДОДАТОК Б

ТЕКСТ ПРОГРАМИ ДЛЯ ЗАГРУЗКИ МОДУЛІВ

```
public static class GenericPluginLoader<T>
    {
        public static ICollection<T> LoadPlugins(string path)
        {
            string[] dllFileName = null;

            if (Directory.Exists(path))
            {
                dllFileName = Directory.GetFiles(path, "*.dll");

                ICollection<Assembly> assemblies = new
                                        List<Assembly>(dllFileName.Length);

                foreach (var dllFile in dllFileName)
                {
                    AssemblyName an = AssemblyName.GetAssemblyName(dllFile);
                    Assembly assembly = Assembly.Load(an);
                    assemblies.Add(assembly);
                }
                try
                {
                    Type pluginType = typeof(T);

                    ICollection<Type> pluginTypes = new List<Type>();

                    foreach (var assembly in assemblies)
                    {

                        if (assembly != null)
                        {
                            Type[] types = assembly.GetTypes();

                            foreach (var type in types)
                            {

                                if (type.IsAbstract || type.IsAbstract)
                                {
                                    continue;
                                }
                                else
                                {

                                    if (type.GetInterface(pluginType.FullName)
                                                                    != null)
                                    {
                                        pluginTypes.Add(type);
                                    }
                                }
                            }
                        }
                    }

                    ICollection<T> plugins = new List<T>(pluginTypes.Count);

                    foreach (var type in pluginTypes)
                    {
```

```
                    var plugin = (T)Activator.CreateInstance(type);
                    plugins.Add(plugin);
                }
                if (plugins != null)
                    return plugins;
                else
                {
                    return null;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }
        return null;
    }
}
```

ДОДАТОК В

ТЕКСТ ПРОГРАМИ ДЛЯ ПЕРЕВІРКИ КРИТЕРІЯ ЗУПИНКИ КЛАСТЕРИЗАЦІЇ

```
private static bool CheckStopCriteria(List<Centroid> prevClusterCenter,
List<Centroid> newClusterCenter, ref int iterationNum)
        {
            iterationNum++;
            if (iterationNum > 11000)
            {
                return true;
            }
            var changeIndex = new int[newClusterCenter.Count()];
            var index = 0;
            do
            {
                int count = 0;
                if (newClusterCenter[index].ClusterDocuments.Count == 0 &&
prevClusterCenter[index].ClusterDocuments.Count == 0)
                {
                    index++;
                }
                else if (newClusterCenter[index].ClusterDocuments.Count != 0 &&
                          prevClusterCenter[index].ClusterDocuments.Count != 0)
                {
                    for (int j = 0; j <newClusterCenter[index]
                                       .ClusterDocuments[0].TermsVector.Count(); j++)
                    {
                      if(newClusterCenter[index].ClusterDocuments[0].TermsVector[j]==
                    prevClusterCenter[index].ClusterDocuments[0].TermsVector[j])
                        {
                            count++;
                        }
                    }
                    if (count ==
                    newClusterCenter[index].ClusterDocuments[0].TermsVector.Count())
                    {
                        changeIndex[index] = 0;
                    }
                    else
                    {
                        changeIndex[index] = 1;
                    }
                    index++;
                }
                else
                {index++;}

            } while (index < newClusterCenter.Count());

                   var stopCriteria = !changeIndex.Where(s => (s != 0))
                                                  .Select(r => r).Any();
            return stopCriteria; }
```

ДОДАТОК Г

ТЕКСТ ПРОГРАМИ ДЛЯ МОДУЛЯ РОЗСИЛКИ SMS

```
public class SmsSender : IMessageSendingPlugin
    {
        private const int ResponseTimeout = 5000;
        private readonly SerialPort _port;
        private readonly AutoResetEvent _receiveNow;

        public SmsSender()
        {
            var config =
ConfigurationManager.OpenExeConfiguration(GetType().Assembly.Location);

            _port = new SerialPort
            {
                PortName = config.AppSettings.Settings["ComPort"].Value,
                Encoding = Encoding.GetEncoding("iso-8859-1"),
                DtrEnable = true,
                RtsEnable = true
            };

            _port.DataReceived += PortDataReceived;
            _port.Open();

            _receiveNow = new AutoResetEvent(false);
        }

        public void Dispose()
        {
            if (_port != null && _port.IsOpen)
            {
                _port.Close();
            }
        }

        private void PortDataReceived(object sender, SerialDataReceivedEventArgs e)
        {
            if (e.EventType == SerialData.Chars)
            {
                _receiveNow.Set();
            }
        }

        public OperationResult SendMessage(string recipientInfo, string subject,
string message,
            TextType typeOfText)
        {
            if (!ValidatePhoneNumber(recipientInfo))
            {
                return new OperationResult(false, "Receiver's phone number is not
valid");
            }

            var result = message.Length <= 70
                ? SendShortSms(recipientInfo, message)
                : SendLongSms(recipientInfo, message);
```

```
            return result == "OK" ? new OperationResult(true, string.Empty) : new
OperationResult(false, result);
        }

        public SendingModule GetPluginInfo()
        {
            return new SendingModule
            {
                Name = "Kyivstar GSM Modem",
                SupportedTextTypes = TextType.PlainText,
                Type = SendingModuleType.Sms
            };
        }

        private bool ValidatePhoneNumber(string number)
        {
            if (number.StartsWith("+380") && number.Length == 13)
            {
                return number.Substring(1).All(char.IsDigit);
            }
            return false;
        }

        private string SendShortSms(string phoneNumber, string message)
        {
            try
            {
                ExecCommand(_port, "AT", ResponseTimeout);
                ExecCommand(_port, "AT+CMGF=0", ResponseTimeout);

                var phoneNumberWithoutPlus = phoneNumber.Substring(1);
                var      encodedPhoneNumber      =      string.Format("0100{0}91{1}",
phoneNumberWithoutPlus.Length.ToString("X2"),
                    EncodePhoneNumber(phoneNumberWithoutPlus));
                var encodedMessage = StringToUcs2(message);
                var                      lengthOfEncodedMessage                      =
(encodedMessage.Length/2).ToString("X2");
                var   command   =   string.Format("{0}0008{1}{2}",   encodedPhoneNumber,
lengthOfEncodedMessage, encodedMessage);
                var lengthOfCommand = Math.Ceiling((double) command.Length/2);

                ExecCommand(_port, "AT+CMGS=" + lengthOfCommand, ResponseTimeout);
                var   receivedData   =   ExecCommand(_port,   "00"   +   command   +
char.ConvertFromUtf32(26), ResponseTimeout);

                return receivedData.EndsWith("\r\nOK\r\n") ? "OK" : "Error";
            }
            catch (Exception exception)
            {
                return exception.Message;
            }
        }

        private string SendLongSms(string phoneNumber, string message)
        {
            try
            {
                ExecCommand(_port, "AT", ResponseTimeout);
                ExecCommand(_port, "AT+CMGF=0", ResponseTimeout);

                var phoneNumberWithoutPlus = phoneNumber.Substring(1);
```

```
                var tp_mr = "00";
                var encodedPhoneNumber = string.Format("41{0}{1}91{2}", tp_mr,
                    phoneNumberWithoutPlus.Length.ToString("X2"),
EncodePhoneNumber(phoneNumberWithoutPlus));

                var  numberOfMessages  =  (int)(message.Length%67  ==  0  ?  (double)
message.Length/67 : Math.Ceiling((double) message.Length/67));

                var randomReferenceNumber = new Random().Next(255).ToString("X2");

                var receivedData = string.Empty;

                for (var i = 0; i < numberOfMessages; i++)
                {
                    var udh = "050003";
                    var partOfMessage = i == numberOfMessages - 1
                        ? message.Substring(i*67)
                        : message.Substring(i*67, 67);

                    var encodedMessage = StringToUcs2(partOfMessage);
                    var    lengthOfEncodedMessage    =    (partOfMessage.Length*2    +
6).ToString("X2");
                    udh += randomReferenceNumber;
                    udh += numberOfMessages.ToString("00");
                    udh += (i + 1).ToString("00");

                    var       command       =       string.Format("{0}0008{1}{2}{3}",
encodedPhoneNumber, lengthOfEncodedMessage, udh,
                        encodedMessage);
                    var        lengthOfCommand        =        Math.Ceiling((double)
command.Length/2).ToString(CultureInfo.InvariantCulture);

                    ExecCommand(_port,       "AT+CMGS="       +       lengthOfCommand,
ResponseTimeout);
                    receivedData    =    ExecCommand(_port,    "00"    +    command    +
char.ConvertFromUtf32(26), ResponseTimeout);
                }

                return receivedData.EndsWith("\r\nOK\r\n") ? "OK" : "Error";
            }
            catch (Exception exception)
            {
                return exception.Message;
            }
        }

        private   string   ExecCommand(SerialPort   port,   string   command,   int
responseTimeout)
        {
            port.DiscardOutBuffer();
            port.DiscardInBuffer();
            _receiveNow.Reset();
            port.Write(command + "\r");

            var input = ReadResponse(port, responseTimeout);
            if   ((input.Length   ==   0)   ||   ((!input.EndsWith("\r\n>   "))   &&
(!input.EndsWith("\r\nOK\r\n"))))
            {
                throw new ApplicationException("No success message was received.");
            }
            return input;
```

```
        }

        private string ReadResponse(SerialPort port, int timeout)
        {
            var buffer = string.Empty;
            do
            {
                if (_receiveNow.WaitOne(timeout, false))
                {
                    var t = port.ReadExisting();
                    buffer += t;
                }
                else
                {
                    if (buffer.Length > 0)
                    {
                        throw   new   ApplicationException("Response   received   is
incomplete.");
                    }
                    throw new ApplicationException("No data received from modem.");
                }
            } while (!buffer.EndsWith("\r\nOK\r\n") && !buffer.EndsWith("\r\n> ") &&
!buffer.EndsWith("\r\nERROR\r\n"));
            return buffer;
        }

        private string EncodePhoneNumber(string number)
        {
            number = number.Trim('+');
            if (number.Length%2 == 1)
            {
                number = string.Format("{0}F", number);
            }
            var result = string.Empty;
            for (var i = 0; i < number.Length; i += 2)
            {
                result   =   string.Format("{0}{1}{2}",   result,   number[i   +   1],
number[i]);
            }
            return result;
        }

        private string StringToUcs2(string str)
        {
            var ue = new UnicodeEncoding();
            var ucs2 = ue.GetBytes(str);

            var i = 0;
            while (i < ucs2.Length)
            {
                byte b = ucs2[i + 1];
                ucs2[i + 1] = ucs2[i];
                ucs2[i] = b;
                i += 2;
            }
            return BitConverter.ToString(ucs2).Replace("-", "");
        }
    }
```

ДОДАТОК Д

КОПІЇ ПУБЛІКАЦІЇ

УДК 519.816

ОПТИМІЗАЦІЯ ЗАДАЧІ РОЗСИЛАННЯ ЕЛЕКТРОННОЇ КОРЕСПОНДЕНЦІЇ

Поворозник В.С.
Тернопільський національний економічний університет, магістрант

І. Постановка проблеми

З задачею розсилання електронної кореспонденції люди стикаються дуже часто, для популяризації свого бізнесу, для автоматизації робочого процесу і т.д. Оскільки розсилання електронної кореспонденції не є безкоштовним процесом, то постає питання в ефективному розподіленні отримувачів цієї кореспонденції, для того щоб вона була надіслана тільки тим кому це насправді потрібно.

Вирішити дану задачу допоможе одна із задач інтелектуального аналізу даних (Data Mining) – кластеризація.

ІІ. Мета роботи

Мета роботи полягає у створенні програмного забезпечення, яке на основі застосування одного із методів кластерного аналізу розподілятиме користувачів та електронну кореспонденцію на групи і в залежності від розподілу груп буде надсилати повідомлення.

ІІІ. Реалізація завдання

Для реалізації завдання обрано алгоритм ієрархічної кластеризації, знизу-вверх (агломеративний алгоритм), а саме алгоритм Ланса-Вільямса.

Спочатку кожен об'єкт вважається окремим кластером. Для кластерів визначається функція відстані:

$$R(\{x\},\{x'\}) = p(x,x') \tag{1}$$

Потім запускається процес злиття. На кожній ітерації замість пари самих близьких кластерів U і V утворюється новий кластер $W = U \cup V$. Відстань від нового кластера W до будь-якого іншого кластера S обчислюється по відстанях $R(U,V), R(U,S)$ і $R(V,S)$, які до цього моменту вже повинні бути відомі:

$$R(U \cup V, S) = \alpha_U R(U,S) + \alpha_V R(V,S) + \beta R(U,V) + \gamma\left|R(U,S) - R(V,S)\right| \tag{2}$$

Загальний алгоритм виглядає так:

1: Ініціалізація кластерів C_1:

$$t := 1;\ C_t = \{\{x_1,...,x_\ell\}\}; \tag{3}$$

2: для всіх $t = 2,...,\ell$ (t-номер ітерації)

3: знайти в C_{t-1} два найближчих кластера:

$$(U,V) := \arg\min_{U \neq V} R(U,V); \tag{4}$$

$$R_t := R(U,V); \tag{5}$$

4: вилучити кластери U і V, добавити утворений новий кластер $W = U \cup V$:

$$C_t := C_{t-1} \cup \{W\} \setminus \{U,V\}; \tag{6}$$

5: для всіх $S \in C_t$

6: вичислити відстань $R(W,S)$ по формулі Ланса-Вільямса

Для обчислення відстані обрано формулу відстані дальнього сусіда:

$$R(W,S) = \max_{w \in W, s \in S} p(w,s);\ \ \alpha_U = \alpha_V = \frac{1}{2}, \beta = 0, \gamma = \frac{1}{2}. \tag{7}$$

Об'єктами виступають ключові слова, що містяться у даних отримувачів, і у самій кореспонденції. Спочатку кластеризуються дані кореспонденції, після чого дані отримувачів, далі відбувається злиття і виділення спільних кластерів, тоді відповідно до кластерів відбувається надсилання кореспонденції. У системі будуть передбачені методи для перегляду/редагування даних

отримувачів, і надісланих об'єктів кореспонденції, буде можливість завантаження даних з файлів. Також будуть зберігатись усі дані про надсилання.

Висновок

Запропоновано застосувати алгоритм ієрархічної кластеризації для оптимізації задачі розсилання електронної кореспонденції, а саме розбиття отримувачів і кореспонденцію на групи (кластери), знаходження спільних груп і в результаті розсилання кореспонденції тільки тим хто найбільш зацікавлений у ній.

Ця система дозволить краще оптимізувати процес розповсюдження інформації в колах користувача системи.

Список використаних джерел

1. Лекции по алгоритмам кластеризации и многомерного шкалирования [Електронний ресурс] //К. В. Воронцов - 21.12.2007. - 18 с. Режим доступу до ресурсу: http://www.ccas.ru/voron/download/Clustering.pdf.
2. Обзор алгоритмов кластеризации данных [Електронний ресурс] – 2010 - Режим доступу до ресурсу: https://habrahabr.ru/post/101338/
3. Применение методов кластеризации для обработки новостного потока [Електронний ресурс] – Режим доступу до ресурсу: http://www.moluch.ru/conf/tech/archive/2/207/

УДК 004.62

ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ПРОЦЕСУ ЗАПОВНЕННЯ СХОВИЩА ДАНИХ ДЛЯ СИСТЕМИ ТАЙМ МЕНЕДЖМЕНТУ

Пойдич В.С.[1)], Струбицька І.П.[2)]

Тернопільський національний економічний університет

[1)] магістрант; [2)] к.т.н., доцент

І. Постановка проблеми

Тайм менеджмент – один з найважливіших критеріїв успішності у бізнесі. «Час це гроші», - не даремно так кажуть. Тому в період новітніх технологій необхідно максимально автоматизовувати процеси, які більшою мірою є марудними при ручному виконанні, але які є істотними для успішності в різних сферах діяльності. Розподіл часу використовується всюди: від простого нагадування про те що і коли треба зробити, до потужного графіку, від якого залежить успішність підприємства. І як вже було сказано, заповнення сховища даних іноді є довгим процесом, що безпосередньо потребує ще часу, який би міг бути використаний на інші важливіші справи. Було розроблено багато методологій по правильному менеджменті часу, як і коли отримувати найкращий результат, як виділяти категорії, як все ж таки встигати і керувати бізнесом і проводити час з сім'єю [1-3].

ІІ. Мета роботи

Метою дослідження є розробка алгоритму, який буде виконувати функцію ефективного розподілу часу відносно певних критеріїв, які можуть бути задані користувачем або визначені системою, і відповідно до результатів будувати максимально гнучкий графік виконання поставлених задач. Також система буде сама навчатись відносно реакцій користувача.

ІІІ. Особливості програмної реалізації продукту

Система буде реалізована на мові C# з використанням технології Xamarin [4], оскільки вона буде орієнтована на мобільні платформи з ОС Android або iOS. Програмна система буде розроблена з використанням алгоритмів, які будуть активно слухати і запам'ятовувати вибори користувача, відносно однієї або іншої категорій. При розробці, як основний критерій, було вирішено використати вже готове розділення Ейзенхауера. Логіка полягає в тому що завдання діляться на 4 категорії:

1. Термінові-важливі;
2. Термінові-неважливі;
3. Нетермінові-важливі;
4. Нетермінові – неважливі.

Саме такий розподіл дозволяє найкраще відобразити все сховище даних, і краще аналізувати дані та подавати користувачу найвідповідніші задачі. На рис. 1 показано куб Ейзенхауера.