SILESIAN UNIVERSITY OF TECHNOLOGY

FACULTY OF AUTOMATIC CONTROL, ELECTRONICS AND COMPUTER SCIENCE

Engineer thesis

Design and implementation of natural language sensitive data extraction tool

author: Maksym Brzęczek

supervisor: Błażej Adamczyk, DSc PhD

consultant: Michał Kawulok, PhD

Gliwice, December 2019

załącznik nr 2 do zarz. nr 97/08/09

Oświadczenie

Wyrażam zgodę / Nie wyrażam zgody* na udostępnienie mojej pracy dyplomowej / rozprawy doktorskiej*.

Gliwice, dnia 25 grudnia 2019

.....
(podpis)

.....
(poświadczenie wiarygodności
podpisu przez Dziekanat)

* podkreślić właściwe

Oświadczenie promotora

Oświadczam, że praca „Design and implementation of natural language sensitive data extraction tool" spełnia wymagania formalne pracy dyplomowej inżynierskiej.

Gliwice, dnia 25 grudnia 2019

.....

(podpis promotora)

Contents

Chapter 1

Introduction

This chapter presents the problem that the project tries to solve and presents the document structure.

1.1 Description of the problem

The invention and propagation of the internet has boosted the ways in which technology impacts all of us. In this age an increasing amount of our everyday life is digitized and dependent on cybernetic systems hosted and operated by independent corporations and institutions. Significant amount of our tasks has become automated through information technology solutions. The phisical word surrounding us also becomes intertwined with technology. We are gradualy connecting things of everyday use like cars or house locks to the web through Internet of Things technologies. This process has made our lives simpler and allowed us to achieve amazing things but it has also made us vulnerable to cybernetic attacks. It is only natural that the rize of the impact of technology was followed by the rise in the cyber crime and cyber security providers [14].

Over the years an ecosystem has emerged that constantly competes with malicious hackers to keep us all secure. One of the elements of this structure is penetration testing also known as ethical hacking. In it's core, this practice is simply simulating a real attack. There are multiple sources that depict approaches used to perform this process. One of the common denominators between all of them is the

importance of gathering information [12]. The reason for that is because the more information you can uncover and analyze, the bigger the chance of finding vulnerable systems or flaws in them. One of the clusters of information in in companies is a communication channel like slack or discord. There are manny situations where employees share information connected to projects and their workplace enviroment. If an attacker was to acces such a platform he could potentially analyse the conversation history in search of sensitive information like ip addresses, logins, passwords, emails, phone numbers, etc.

Such information is especially important from a legal point of view. Introduction of General Data Protection Regulation in 2016 has put a preassure on manny legal bodies to responsibly handle people's personal data under a threat of heavy financial penalties [18]. Monitoring of the data located in the private entity's internal communication channel might prove very useful in fulfilling the legislative requirements of private information processing.

Unfortunately such a task may be very time and resource consuming. A tool capable of scanning the history of communication channel in search of data that would meet some established criteria could however fulfill this job or at least increase efficiency of the person responsible for it.

1.2 Project scope

This thesis focuses on the research into proper implementation of a cyber security tool with the purpose of processing a large amount of natural language messages in search of data that can be classified as personal or sensitive from the cyber security point of view. Special focus is put on possibility of calibration and customisation of the search engine and it's reusability, regardless of circumstances and enviroment.

The project will primarily focus on finding nine categories of information:

- IP addresses - numerical label assigned to each device connected to a computer network that uses the Internet Protocol for communication [16].
- National identification numbers - numbers issued by a Government Entity as a means of providing Identification of their citizens and or Aliens[19].

- ID card numbers - serial numbers of documents (such as a cards) bearing identifying information about and often a photograph of the individual whose name appears on it[3].
- MAC addresses - unique identifier for an Ethernet or network adapter over a network. It distinguishes different network interfaces and is used for a number of network technologies, particularly most IEEE 802 networks, including Ethernet [8].
- Domain names - combinations of letters and numbers used in combination of the various domain name extensions, such as .com, .net to find and identify computers on the Internet[10].
- Email addresses - serieses of letters, numbers, and symbols used to send and receive email [1].
- Passwords - secret words or combinations of letters or numbers, used for communicating with another person or with a computer to prove who you are [2].
- Usernames - a unique sequence of characters used by a person with access to a computer, network, or online service [5].
- Phone numbers - numbers assigned to a telephone line for a specific phones or set of phones (as for a residence) that are used to call those phones [4].
- Additional - data types specified and implemented by the users.

1.3 Description of chapters

This document is divided into seven chapters with following focus:

- Chapter 1 Introduction - Presentation of the problem domain and scope, introduction of the document structure.
- Chapter 2 Problem analysis - Research of the topic and design of the program.

- Chapter 3 Requirements and tools - Description of the used technologies and required prerequisites.
- Chapter 4 External specification - Instruction of the program usage.
- Chapter 5 Internal specification - Elaboration on the programs internal structure.
- Chapter 6 Verification and validation - Presentation of the testing methodology and results.
- Chapter 7 Conclusions - Final remarks and conclusions for the future of the project.

Chapter 2

Problem analysis

2.1 Existing solutions

There are not manny widely available solutions that are capable of performing the job presented in the thesis introduction. However the scope of the project shares similarities with Data Lekage Prevention Systems which focus on analysis of the content of confidential data and the surrounding context in order to prevent unwanted disclosures of information. Those programs usually use up to three techniques to monitore the sensitive information - regular expressions (regexes), data fingerprinting and statistical analysis [11]. This approach could be adapted for the purposes of this thesis.

Regular expressions are an abstraction of key-word search that enables the identification of text using apattern instead of an exact string. They are a tool frequently used for parsing users input and capturing parts of strings [13]. Regexes are used in Data Leakage Prevention Systems for detecting data like credit card numbers and social security mumbers [11] which belong to the scope of this thesis. It might be possible to extend this approach in to other categories of data covered by the designed program.

Regular expressions are however known to have high false positives rates [11]. It might be advantageous for the purposes of this project to take under consideration additional properties of the considered data types. It is possible for the personal information issued by some entities to implement a check sum algorithm used

for validation of authenticity. Implementing a adjustable additional check of the data discovered by regular expressions could possibly reduce the amount of false positives. Such feature could be utilised by individual user to further enhance accuracy in any case where identifing additional patterns, that escape regular expression domain, is possible.

While dictionary search does not seem to be a common part of Data Leakage Prevention Systems it might prove useful in achieving the goal of this thesis. The natural language messages processed by the developed program may contain keywords indicating presence of the desired data, which might have not been detected by the initial regex search. Implementing a dictionary search which focuses on words like "password", "login" etc. could potentially increase the reliability of the solution.

2.2 Modularity and Customisation

Each usecase of the designed system might differ depending on the user, enviroment and multitude of other conditions. Some clients of the program might be interested in searching for a subset of data types implemented in the program. It is also possible that an unanticipated in the design information type might be of a particular interest for a user.

In order to mitigate those problems a modular approach to the design of program could be taken. Its main goal would be to allow specifing which of the stock implemented data types to search for as well as simplifying the augmentation of the program's scope.

It is also crucial to take under consideration the fact that the structure of some data types can differ significantly between uses. An example of this is a personal identification number which may vary depending on issuing authority. Giving the user a easily accessible possibility to adjust the regular expressions used in searching process as well as addictional check functions may increase the use cases of the solutions.

2.3 Result presentation

While the focus of this thesis is a program which focuses on finding some sensitive data, it might be useful to put a special emphasis on the presentation of the results. Information discovered by the solution might be riddled with false positives. In such case giving the user a possibility to easily analyse the message and conversation context of found data could increase the usefulness of the tool.

2.4 Enviroment independance

The typical user of the developed program might need to run it on multiple different operating systems. Administrators might require the program to run on Windows operating system while IT security specialists could possibly want to deploy it on a Kali Linux which is a Debian-based Linux distribution aimed at advanced Penetration Testing and Security Auditing. Developing a program capable of runing regardless of the enviroment in which it is used would possibly allow for a bigger user base.

2.5 Input variability

The program is supposed to be capable of operating on manny different types of input like emails, Facebook messages, Slack messages etc. In order to allow for that it may be necessary to design a arbitrary input format. A user would have to be required to transform initial data in their possession to fit this predefined standard.

Chapter 3

Requirements and tools

3.1 Tools

In order to achieve the goal of this thesis following tools were used:

- Visual Studio Code - a source-code editor developed by Microsoft for Windows, Linux and macOS. It's source code is free and open source and released under the permissive MIT License. It was used with a Python plugin which provides a rich support for the Python language , including features such as IntelliSense, linting, debugging, code navigation, code formatting, Jupyter notebook support, refactoring, variable explorer, test explorer, snippets, and more [15][7].
- Qt Designer - tool for designing and building graphical user interfaces (GUIs) with Qt Widgets. Allows for composing and customization of windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and testing of them using different styles and resolutions[9].

3.2 Technologies

Entire project is developed with use of following technologies:

- Python 3.7.3 - an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to

object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms[17]. There are multiple arguments that support choosing it as privary development language for the project. It has implemented modules for regular expression search and graphical user interface. The scriptive nature of it makes it easy to create programs that are capable of runing in any operating system that has Pyhton interpreter installed.

- JSON - a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language[6].

3.3 Requirements

The program designed in this thesis is supposed to perform following tasks:

- Perform a regex search on natural language message sets.
- Additionally check data found with regular expressions with false positives check function if one is available.
- Perform dictionary search on natural language message sets.
- Display the found data.
- Allow for displaying of message context of found data.
- Allow for implementation of custom searches.
- Save to and load results from a output file.

The program designed in this thesis is supposed to be capable of runing regardless of operating system.

3.4 Use cases

Chapter 4

External specification

4.1 Requirements

For the program to work correctly, some software requirements have to be meet. It was developed with Python and requires it's interpreter in version at least 3.7.3 to be installed. Additionally following Python packages have to be installed:

- pyqt5 - used in GUI implementation.
- json - utilised for data input and output.
- os - used or enviroment related actions like proper joining of paths.
- traceback - required for proper error presentation.
- importlib - allows for dynamic loading of the additional check functions.
- threading - implements Python multi-thread operations.
- re - package for working with Python regular expressions.

Some of those are contained within standard Python installation and the rest can be obtained installed with package-management system called pip. Example of installation script on Windows 4.1 and Linux 4.2 that fulfill the above requirements.

```
1    python -m pip install pyqt5
```

Figure 4.1: Windows installation script

```
1        !/bin/bash
2        python -m pip install pyqt5
```

Figure 4.2: Linux installation script

4.2 User manual

4.3

Chapter 5

Internal specification

- concept of the system
- system architecture
- description of data structures (and data bases)
- components, modules, libraries, resume of important classes (if used)
- resume of important algorithms (if used)
- details of implementation of selected parts
- applied design patterns
- UML diagrams

Use special environment for inline code, eg **descriptor** or **descriptor_gaussian**. Longer parts of code put in the figure environment, eg. code in Fig. 5.1. Very long listings–move to an appendix.

```
1  class descriptor_gaussian : virtual public descriptor
2  {
3      protected:
4          /** core of the   set */
5          double _mean;
6          /** fuzzyfication of the gaussian fuzzy set */
7          double _stddev;
8
9      public:
10         /** @param mean core of the set
11             @param stddev standard deviation */
12         descriptor_gaussian (double mean, double stddev);
13         descriptor_gaussian (const descriptor_gaussian & w
               );
14         virtual ~descriptor_gaussian();
15         virtual descriptor * clone () const;
16
17         /** The method elaborates membership to the
               gaussian fuzzy set. */
18         virtual double getMembership (double x) const;
19
20 };
```

Figure 5.1: The **descriptor_gaussian** class.

Chapter 6

Verification and validation

- testing paradigm (eg V model)
- test cases, testing scope (full / partial)
- detected and fixed bugs
- results of experiments (optional)

Chapter 7

Conclusions

- achieved results with regard to objectives of the thesis and requirements
- path of further development (eg functional extension ...)
- encountered difficulties and problems

Table 7.1: A caption of a table is **above** it.

	method						
			alg. 3			alg. 4, $\gamma = 2$	
ζ	alg. 1	alg. 2	$\alpha = 1.5$	$\alpha = 2$	$\alpha = 3$	$\beta = 0.1$	$\beta = -0.1$
0	8.3250	1.45305	7.5791	14.8517	20.0028	1.16396	1.1365
5	0.6111	2.27126	6.9952	13.8560	18.6064	1.18659	1.1630
10	11.6126	2.69218	6.2520	12.5202	16.8278	1.23180	1.2045
15	0.5665	2.95046	5.7753	11.4588	15.4837	1.25131	1.2614
20	15.8728	3.07225	5.3071	10.3935	13.8738	1.25307	1.2217
25	0.9791	3.19034	5.4575	9.9533	13.0721	1.27104	1.2640
30	2.0228	3.27474	5.7461	9.7164	12.2637	1.33404	1.3209
35	13.4210	3.36086	6.6735	10.0442	12.0270	1.35385	1.3059
40	13.2226	3.36420	7.7248	10.4495	12.0379	1.34919	1.2768
45	12.8445	3.47436	8.5539	10.8552	12.2773	1.42303	1.4362
50	12.9245	3.58228	9.2702	11.2183	12.3990	1.40922	1.3724

Bibliography

[1] Definition of a email address in cambridge english dictionatry. https://dictionary.cambridge.org/dictionary/english/email-address. [access date: 2019-12-23].

[2] Definition of a password in cambridge english dictionatry. https://dictionary.cambridge.org/pl/dictionary/english/password. [access date: 2019-12-23].

[3] Definition of id by merriam-webster. https://www.merriam-webster.com/dictionary/id. [access date: 2019-12-23].

[4] Definition of telephone number by merriam-webster. https://www.merriam-webster.com/dictionary/telephone%20number. [access date: 2019-12-23].

[5] Definition of username at dictionary.com. https://www.dictionary.com/browse/username. [access date: 2019-12-23].

[6] Json. https://www.json.org/json-en.html. [access date: 2019-12-23].

[7] License - visual studio code. https://code.visualstudio.com/license. [access date: 2019-12-23].

[8] Media access control address (mac address). https://www.techopedia.com/definition/5301/media-access-control-address-mac-address. [access date: 2019-12-23].

[9] Qt designer manual. https://doc.qt.io/qt-5/qtdesigner-manual.html. [access date: 2019-12-23].

[10] What is a domain name? `https://www.website.com/beginnerguide/domainnames/8/1/What-is-a-domain-name?.ws`. [access date: 2019-12-23].

[11] Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. A survey on data leakage prevention systems. *Journal of Network and Computer Applications*, 62:137–152, 2016.

[12] Rafay Baloch. *Ethical Hacking and Penetration Testing Guide*. Auerbach Publications, 2014.

[13] Kathryn T. Stolee Carl Chapman. Exploring regular expression usage and context in python. In *25th International Symposium on Software Testing and Analysis*, pages 282–293, 2016.

[14] Rajesh Kumar Goutam. Importance of cyber security. *International Journal of Computer Applications*, 111(7):4, 2016.

[15] Frederic Lardinois. Microsoft launches visual studio code, a free cross-platform code editor for os x, linux and windows. `https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-c` [access date: 2019-12-23].

[16] Name Surname and Name Surname. Internet protocol. *DARPA Internet Program Protocol Specification*, 1981.

[17] Guido van Rossum and the Python development team. *PythonTutorial Release3.8.1*. Python Software Foundation, 2019.

[18] Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer International Publishing, Cham, 2017.

[19] Jim Willeke. National identification number. `https://ldapwiki.com/wiki/National%20Identification%20Number`. [access date: 2019-12-23].

Appendices

List of abbreviations and symbols

DNA	deoxyribonucleic acid
MVC	model–view–controller
N	cardinality of data set
μ	membership function of a fuzzy set
$\mathbb{E}$	set of edges of a graph
$\mathcal{L}$	Laplace transformation

Listings

(Put long listings in the appendix.)

```
1  partition fcm_possibilistic::doPartition
2                                  (const dataset & ds)
3  {
4      try
5      {
6          if (_nClusters < 1)
7              throw std::string ("unknown␣number␣of␣clusters"
                   );
8          if (_nIterations < 1 and _epsilon < 0)
9              throw std::string ("You␣should␣set␣a␣maximal␣
                   number␣of␣iteration␣or␣minimal␣difference␣--
                   ␣epsilon.");
10         if (_nIterations > 0 and _epsilon > 0)
11             throw std::string ("Both␣number␣of␣iterations␣
                   and␣minimal␣epsilon␣set␣--␣you␣should␣set␣
                   either␣number␣of␣iterations␣or␣minimal␣
                   epsilon.");
12
13         auto mX = ds.getMatrix();
14         std::size_t nAttr = ds.getNumberOfAttributes();
15         std::size_t nX    = ds.getNumberOfData();
16         std::vector<std::vector<double>> mV;
17         mU = std::vector<std::vector<double>> (_nClusters)
```

```
        ;
18      for (auto & u : mU)
19          u = std::vector<double> (nX);
20      randomise(mU);
21      normaliseByColumns(mU);
22      calculateEtas(_nClusters, nX, ds);
23      if (_nIterations > 0)
24      {
25          for (int iter = 0; iter < _nIterations; iter++)
26          {
27              mV = calculateClusterCentres(mU, mX);
28              mU = modifyPartitionMatrix (mV, mX);
29          }
30      }
31      else if (_epsilon > 0)
32      {
33          double frob;
34          do
35          {
36              mV = calculateClusterCentres(mU, mX);
37              auto mUnew = modifyPartitionMatrix (mV, mX);
38
39              frob = Frobenius_norm_of_difference (mU,
                    mUnew);
40              mU = mUnew;
41          } while (frob > _epsilon);
42      }
43      mV = calculateClusterCentres(mU, mX);
44      std::vector<std::vector<double>> mS =
            calculateClusterFuzzification(mU, mV, mX);
45
46      partition part;
47      for (int c = 0; c < _nClusters; c++)
```

```
48          {
49              cluster cl;
50              for (std::size_t a = 0; a < nAttr; a++)
51              {
52                  descriptor_gaussian d (mV[c][a], mS[c][a]);
53                  cl.addDescriptor(d);
54              }
55              part.addCluster(cl);
56          }
57          return part;
58      }
59      catch (my_exception & ex)
60      {
61          throw my_exception (__FILE__, __FUNCTION__,
                __LINE__, ex.what());
62      }
63      catch (std::exception & ex)
64      {
65          throw my_exceptionn (__FILE__, __FUNCTION__,
                __LINE__, ex.what());
66      }
67      catch (std::string & ex)
68      {
69          throw  (__FILE__, __FUNCTION__, __LINE__, ex);
70      }
71      catch (...)
72      {
73          throw my_exception (__FILE__, __FUNCTION__,
                __LINE__, "unknown␣expection");
74      }
75  }
```

Contents of attached CD

The thesis is accompanied by a CD containing:

- thesis (LaTeX source files and final `pdf` file),
- source code of the application,
- test data.

List of Figures

List of Tables