UNIVERSITY OF TWENTE

MASTER THESIS

Verification of User Information

Author:
Marco SCHULTEWOLTER

Graduation commitee:
Dr. Ir. Djoerd HIEMSTRA
Prof. Dr. Ir. Boudewijn R. HAVERKORT
Dr. Ir. Maurice VAN KEULEN
Franz-Josef LEUDERS

Faculty of Electrical Engineering, Mathematics and Computer Science
(EEMCS)

July 2016

"You can have data without information, but you cannot have information without data."

Daniel Keys Moran

UNIVERSITY OF TWENTE

Abstract

Master Thesis

Verification of User Information

by Marco SCHULTEWOLTER

Often, software providers ask users to insert personal data in order to grant them the right to use their software. These companies want the user profile as correct as possible, but users sometimes tend to enter incorrect information. This thesis researches and discusses approaches to automatically verify this information using third-party web resources.

Therefore, a series of experiments is done. One experiment compares different similarity measures in the context of a German phone book directory for again different search approaches. Another experiment takes the approach to use a search engine without a specific predefined data source. Ways of finding persons in search engines and of extracting address information from unknown websites are compared in order to do so.

It is shown, that automatical verification can be done to some extent. The verification of name and address data using external web resources can support the decison with Jaro-Winkler as similarity measure, but it is still not solid enough to only rely on it. Extracting address information from unknown pages is very reliable when using a sophisticated regular expression. Finding persons on the internet should be done by using just the full name without any additions.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many companies provide services to customers they do not know personally. Especially Software as a Service (SaaS) or in general services that are provided via the internet are prone for such scenarios.

When a company has a business relationship with a customer, the company is willing to have a customer profile containing complete and most notably correct personal data about him or her. Often money plays also an important role in such services, at least when it comes to the payment part of the services. Then trust in the customers is very important. Having correct customer information not only helps to have a transparent overview about the money flow, but customers are also less motivated for fraudulent actions. This increases the trust in the customers. Experience says that many people fill in incorrect data in sign-up processes or at any other later point of time.

This project is about finding out, whether the data entered by a user is correct or not, following the famous saying "*Trust is good. Control is better*". As basis for this check, mainly publicly available data on the internet should be used. This includes data from social network sites and other web sites like telephone directories or even generic other web pages which contain personal data. This project is not about detecting, whether a user is the one he claims to be, as long as the user he or she claims to be is a real existing person and all information filled in is correct, but to check whether single information items they fill in are correct or not.

1.1 Practical environment

Tobit Software, based in Ahaus (Germany), also suffers from this problem. In this thesis, this company will be used as a specific case for this general problem. Tobit Software is a producer of several commercial off-the-shelf (COTS) software products. One very popular product ist called *David*, an information server for professional use. It handles all sorts of communication for a company, for instance e-mails, phone calls, faxes and even traditional letters.

Another popular product is Chayns, which is offered as a COTS software product. Chanys automatically builds native smartphone

apps for various platforms by taking the basis information from Facebook pages. Next to the Facebook channel, one can add custom content and features to the app. This makes the apps to be created highly customizable, almost as custom software. For the customers it is free to build these apps, this is one reason for the huge amount of more than 100.000 already existing apps.

Since many years, Chayns apps are used to control Tobit Software's own catering locations, including ordering and payment. Now the payment services are getting rolled out to a higher level, so that their customers' apps can also be used to handle their own payments. Therefore, a finetrading service called OPM[1] is launched.

To be able to pay by this OPM service, people need to have a Chayns user account. Such accounts are used on a prepaid base by default. Before a user wants to order and pay an article, he or she needs to have a positive balance with enough money covering the transaction sum. Therefore, users can top up their account in real-time using one of various methods:

- SEPA Direct Debit,
- Credit Card,
- PayPal,
- SofortÜberweisung[2],

Under specific circumstances, e.g., when there have been several successful transactions, the solvency of a user can raise to a higher category. Then the users are granted a credit limit and they can order and pay without topping their account up beforehand. Once a day, the account is balanced by SEPA Direct Debit automatically, when a certain amount is exceeded.

A credit limit with postpaid balancing is always a leap of faith you have to grant a customer. Having millions of not personally known customers leads to a high probability of fraudulent use. In such a case, it can be of benefit to be able to identify the fraudulent user. In order to do so, you have to ensure correct user data.

1.2 Purpose

The purpose of this project is to do research and show the possibilities Tobit Software or any instance in general, has to verify user information using public third-party data sources. The purpose is not

[1] OPM initially was an acronym for "Other People's Money"

[2] SofortÜberweisung is a German payment method triggering a guided bank transfer using internet banking techniques. Since it is a fully guided process, a confirmation of the successful payment can be given immediately.

to implement a fully-working verification system in a production environment, but to show the possibilities with a prototype implementation.

1.3 Research question

The problem statement above leads to the following research question:

> *Can information about a user/person be verified automatically using publicly available third-party web resources?*

The research question is divided into the following subquestions:

1. Where can information of the users be found?
2. What information should be searched for to find persons?
3. How should the found information be compared with given information?

The first subquestions gives us insight into potential data sources where personal data can be retrieved from. The second subquestion tells us how we can find persons at those data sources. Finally, the third subquestion answers, how the information actually should be compared in order to verify personal information.

1.4 Research method

The research question with the subquestions is answered by doing experimentation with a prototype implementation within the practical environment of Tobit Software. The experimentation contains various subexperiments. Firstly, we will compare various similarity measures for the verification in a specific predefined data source.

Then, we investigate different approaches for searching for a person and information extration. Knowing, what approaches work best for that, we do verification using a search engine without any predefined data sources and evaluate it.

How the research is done, is expalained in more detail in Chapters 3 and 5.

1.5 Structure of the thesis

In Chapter 2, the thesis first gives some background information. This includes related work and also some explanation of how other projects in practice solve the problem of verifying user information.

Then, we will discuss some ethical aspects that play a role in the context of this project in Chapter 4. Chapter 3 sketches a global design about a service that could be implemented to solve the problem or at least enables us to do the neccessary experiments, whereas chapter 5 explains how the actual experiment are set up.

After that, you can find the results of the experiment in Chapter 6. Finally, in Chapter 7, the initial research questions are answered in the conclusion.

Chapter 2

Background Information

This chapter will introduce some useful background information for this thesis. It will define personal data and gives some approaches to measure string similarities. Moreover, there are some approaches how other instances do user information verification in practice.

2.1 Personal Data and PII

According to the German Federal Data Protection Act[1], *Personal Data* means any information concerning the personal or material circumstances of an identified or identifiable individual, which is basically the same definition as stated in the European Union (EU) Data Protection Directive[2]. The protection requirements of Personal Data are quite far-reaching in Germany and Europe in general.

A very essential question in finding information about users in external sources is, how to be sure that the given piece of information actually belongs to the user you want to verify. To do so, you need personal data that can identify a person. This specific data is often called *Personally Identifiable Information* (PII). However, the definition of PII varies by content. In legal context, California Senate Bill 1386, for instance, includes Social Security Numbers, driver's license numbers and financlial accounts, while it does not include email addresses and telephone numbers. Most other definitions, however, include such information. Also a person's names can uniquely identify a person, when the name is uncommon.

But even when a single information item is not enough to identify some person, maybe a combination of items is still able to do so. Next to PII, there are quasi-identifiers (or sometimes quasi-PII). A quasi-identifier is information that makes people re-identifiable when combining with other quasi-identifiers.

Some time ago, the concept of removing directly identifiable information was seen as sufficient anonymization, e.g., to publish statistical data from the healthcare sector. Today we know that these combinations can also lead to the ability for so-called *re-identification*

[1] Bundesdatenschutzgesetz (BDSG).

[2] Directive on the protection of individuals with regard to the processing of personal data and on the free movement of such data.

or *de-anonymization*[28, 17]. Information that can be used with other information that needs to be combined to re-identify a person is called a quasi-identifier (or sometimes quasi-PII).

2.2 String similarity

In a situation like given in this problem, a usual string comparison is not sufficient to compare user data. There are various variations of how data is written. The german term *"Straße"* in street names for instance is often abbreviated as *"Str."*, as in *Parallelstr.* instead of *Parallelstraße*. Furthermore some street names are splitted in multiple words separated by a space, while some people use dashes to seperate them. The same issue applies to house numbers, where people often use a notation like *"23a"* instead of the standardized equivalent notation *"23 A"*.

Similar phenomena appear in names. In Germany it is, different than, e.g., in the Netherlands, common to use full first names instead of just the initials. People named *"Karl-Heinz"*, which is officially one double-barreled name, sometimes write their first name as *"Karl Heinz"* or even leave out the second part of the name and use just *"Karl"* as their first name or use it as a merged single name (*"Karlheinz"*). Other people with multiple first names can appear with just a part of their first names in one place and and with all names or abbreaviated middle names in another place (think about *"Reinhard Josef Meyer"* vs *"Reinhard J. Meyer"* vs *"Reinhard Meyer"*).

The last name can also occur in different ways, especially the double-barreled last names, where sometimes one of the names is left out and only one of the two name parts is used. The situation of a fully changed last name e.g. after marriage is, in contrast, nothing that could be handled by techniques indicating string similarity.

The following part will describe some common functions that can be used to detect string similarity.

2.2.1 Jaccard

The **Jaccard coefficient** makes use of so-called n-grams, which indicate sub-strings of length n in longer strings. While a unigram means an n-gram with $n = 1$, there are also bigrams ($n = 2$), trigrams ($n = 3$) and so on. The Jaccard coefficient usually uses the amount of bigrams the two strings have in common and divides it by the size of union bigrams [6].

$$sim_{Jaccard}(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

For instance, let A be the bigram set of "Möller" and B be the bigram set of "Möllers", then A contains "Mö", "öl", "ll", "le" and "er"

and B contains "Mö", "öl", "ll", "le", "er" and "rs". The intersection of both strings are all bigrams except of "rs" from B, which are 5 in total. The size of the union set is 6, so $sim_{Jaccard} = \frac{5}{6} \approx 0,83$.

2.2.2 Levenshtein

Another popular similarity metric is the **Levenshtein distance** (also Edit Distance). It calculates the smallest number of edit operations required to change one string into another [6, 19]. Such operations can be deletions, substitutions and insertions, while each of them has the same costs (1). To get a similarity measure from the distance, you set it in relation with the longest string's length, since the edit distance can never be longer than that.

$$sim_{Levenshtein}(s_1, s_2) = 1 - \frac{dist_{Levenshtein}(s_1, s_2)}{\max(s_1, s_2)}$$

Let in this case s_1 be "Karl-Heinz Meyer" and s_2 be "Karlheinz Meiers", then to change s_1 into s_2 needs one deletion of the dash, one substitution to transform "y" into "i" and one insertion of the "s" at the end of the last name. Calculating the distance in a case-sensitive manner, it would also cost one substitution to transform the "H" into a non-capital letter. Assuming that we want to have no case-sensitive comparision, the costs add up to 3. Then the Levenshtein similarity would be $sim_{Levenshtein} = 1 - \frac{3}{16} \approx 0,81$.

Damerau-Levenshtein

The **Damerau-Levenshtein distance** is a variation of the Levenshtein distance. The difference is that transposing two adjacent characters is calculated as one edit operation instead of the otherwise necessary two operations [6, 7]. Damerau stated, that when adding this case, 80% of all spelling mistakes are covered to just need one edit operation to match the other string [7].

2.2.3 Jaro

A common algorithm for name matching in record-linkage systems is the **Jaro algorithm** which is calculated using the common characters c from both strings and also the the number of transpositions needed to rearrange them t, which is calculated as $\lfloor \frac{|c|}{2} \rfloor$. The characters are only considered as common, if they are within half the length of the longer string, thus not farther than $\left\lfloor \frac{\max(|s_1|,|s_2|)}{2} \right\rfloor - 1$.

Let c be the set of these common characters and t be the amount of transpositions needed, then

$$sim_{Jaro}(s_1, s_2) = \frac{1}{3} \left(\frac{|c|}{|s_1|} + \frac{|c|}{|s_2|} + \frac{|c| - t}{|c|} \right).$$

Jaro-Winkler

With the purpose of name matching in mind, a popular improvement of this algorithm is the **Jaro-Winkler algorithm** [29]. Empirical study has shown that typically less errors occur at the beginning of names [24]. This fact is taken into account with Jaro Winkler by increasing the weight for common initial characters. The set of common initial characters is represented by p.

$$sim_{JaroWinkler}(s_1, s_2) = sim_{Jaro}(s_1, s_2) + \frac{p}{10} * (1 - sim_{Jaro}(s_1, s_2))$$

2.3 Verification in practice

2.3.1 SCHUFA

In Germany, there are multiple credit bureaus, that protect companies from credit risks by estimating the creditworthiness of their customers. The most popular one is SCHUFA[3]. Founded in 1927 [2], SCHUFA has grown to the biggest provider for this purpose, so that it has collected information about 66,3 million people [3], which is approximately equal to the German population older than 18 [4, 15]. The SCHUFA's customers report contracts and granted credits of - on the other hand - their customers. Having all this information, SCHUFA can calculate a creditworthiness score. How it is calculated is a secret.

Havig this knowledge about people, the SCHUFA launched an additional service for checking identities (*SCHUFA-IdentitätsCheck*). This service makes it possible for SCHUFA's customers to check the name and address of new customers. There is also the so-called *SCHUFA-IdentitätsCheck Jugendschutz*, which checks, whether a given person is older than 18 or not. Such identity checks do not influence the score of creditworthiness.

Customers always have to actively agree with a check or a report to the SCHUFA database. Not agreeing, in most cases results in the consequence that one cannot register with the service. The check, of course, only works because of the rather complete database of German citizens.

2.3.2 Facebook

Facebook is the biggest social network in the world with more than a billion daily active users worldwide in the third quartile of 2015. The userbase in Europe is approximately one quarter of that. The tendency is still rising (see Figure 2.1). Users provide their personal

[3]Schutzgemeinschaft für allgemeine Kreditsicherung (Protection company for general creditworthiness)

data by choice to Facebook and make it publicly available in many cases [10].

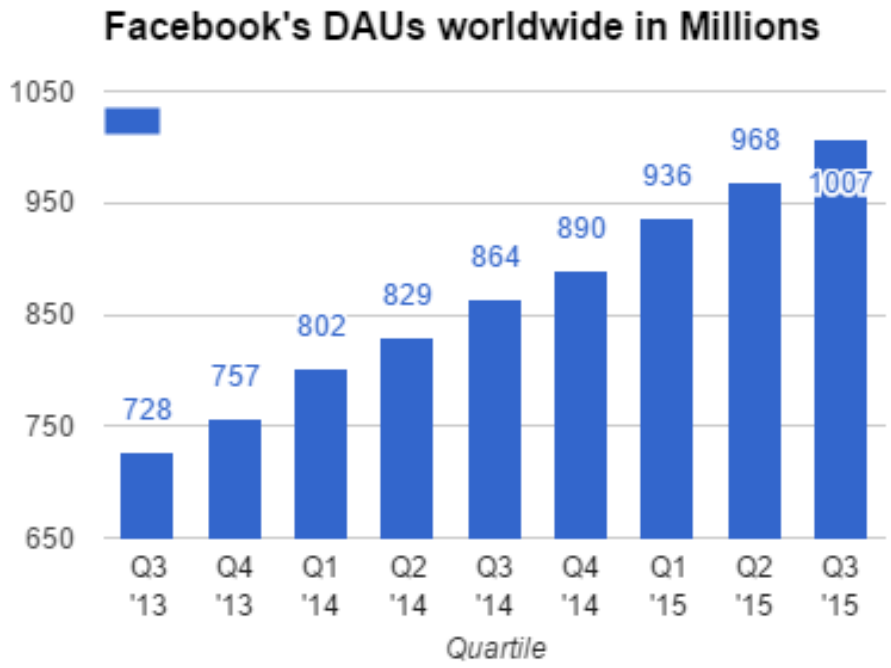


FIGURE 2.1: Daily active users (DAUs) in Facebook worldwide

In 2005, Gross and Acquisti showed in [16], that 89% of all names are realistic in Facebook, which could be reasoned by the real name policy [21]. They name it the *authentic identity*, which they define as the name, the user is called in real life by friends [11]. Facebook disables accounts they identify to use a fake name automatically and forces them to identify via official documents like ID cards or birth certificats. The names mentioned in the accepted documents can conflict with the authentic identity, they claim they need [12]. This is why facebook was criticized by the Electronic Frontier Foundation in an open letter [14, 8] supported by many other signees. For a period of time, Facebook even asked their users actively, whether given names of their friends are their actual real name.

In December 2015, Facebook announced some changes in the names process [9]. In the future, people should add further details when they report someone as a fake name holder. In case of special circumstances, the user with the fake name shall be able to stick to the pseudonym. What these special circumstances are, is not totally clear, but one can think about people who got cyber bullied in the past or something similar. In such a case, Facebook still wants a proof of identity, so that at least Facebook knows who you are.

Apparently, users in most cases agree to send a copy of an indentifing document to facebook and tolerate these extensive identification process. For many users, Facebook is a major communication

channel, they cannnot abstain from.

2.3.3 Socure

There is a commercial project called Socure, which solves a related problem [22]. Socure offers an API, that can authenticate customer identities in realtime to detect identity fraud. They specialize on financial institutions and their customers. Their patented product called *Social Biometrics*™ uses multiple social networks and other online media to calculate an authenticity score for a user account as a whole.

Therefore, the user can authenticate via a social network (Facebook, Twitter, LinkedIn, Google+ or other) and then collects data about your profile, your activities, your applications, your behaviour, your friends, family and followers. Additionally - or instead of that, if there is no social network login - the Socure system checks the data they internally collected in their own graph database. After all, it computes the authenticity score, which is a number between 1 (low degree of authenticity/fake person) and 10 (high degree of authenticity/real person). The actual way to calculate the score is not publicly available. [22]

Chapter 3

Global Design

In this chapter, a global design of a possible verification system is sketched. It discusses some choices within such a system, discusses issues in potential data sources and gives insight into the test set that is used. In addition to that, the modes that will be compared will be expalined. Finally, there is an explanation about how the result of this system should look like.

3.1 Information groups

The information of a person is split into multiple fields in the database. For many purposes it is useful to split an address into street, zip code and city for example. For our specific purposes, this information will be combined into one *information group*. Instead of verifying the street and zip code sperately, the address is seen as a whole. For instance, in case the street is not correct while the city is, the address is seen as not correct in general.

The same applies to the information group called name, where the first and last name are combined. For the phone numbers, all possible phone numbers like mobile, private landline or office phone numbers are combined and are validated as one. If one of those phone number is correct, it is seen as correct and the system should return, that this attribute is correct. This choice also leads to the fact that the different phone types are not be verified on where they are stored. If a private phone number is stored as business phone number this is seen as correct anyway.

3.2 Data sources

This project relies on information third-party data sources provide about the users. The following section discusses different approaches and issues with data sources. At the end there will be an overview, which issues apply to which data sources.

TABLE 3.1: Information groups

Attribute	Information group
first name	name
last name	
street	address
ZIP	
city	
mobile phone	phone
private phone	
business phone	
email	email
birthday	birthday

3.2.1 API vs. Web Scraping

To access data from third-party data sources, there are different approaches. Some providers offer an API to make the information accessible in a machine readable format. Often an API key is needed which can be gained by registering or applying for it. Moreover, usually there are limitations in how you are allowed to use these APIs, including the amount of requests in a specific period of time or the purpose of the applicaton.

An alternative approach is to use web scraping, that is to query for the information like real users would do via the website and then automatically scrape the relevant information from the resulting webpage. Using this approach has less restrictions than using the API, since the requests are seen as being sent from real users.

Generally speaking, an API access is of course the preferred way for retrieving information from third-party data sources, but in many cases this is not possible, so that web scraping then is the only usable alterantive to access this information.

3.2.2 Issues with accessing data

Some web sites so not want programs to access the website. They especially try to prevent automated software from scraping the information from their pages.

robots.txt

Providers of web pages can indicate in a so-called robots.txt, which parts of a website they allow automated robots to scrape and which parts are not. It can also contain rules that are valid for specific robots, such that, e.g., the Google crawler should be able to crawl the page in order to be indexed at the Google web search, but to disallow all other crawlers to do so. The robots.txt does not contain any

hard rules that make it impossible to still scrape the site, but it is recommended to follow them.

CAPTCHAs

A CAPTCHA, which is an acronym for *Completely Automated Public Turing Test to tell Computers and Humans Apart*, is in most cases an image containing text that is obscured in a way that machines should not be able to automatically read it while real human beings are. Sites can ask their users to solve these CAPTCHAs, when they want to prevent others than real users to access information on the page.

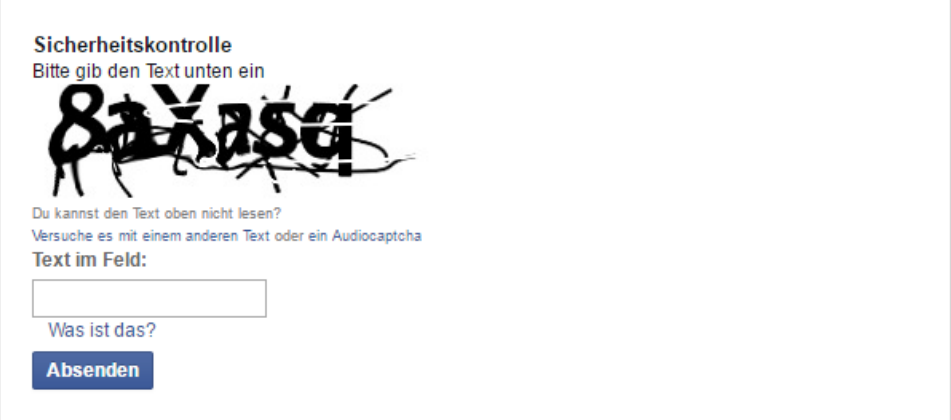


FIGURE 3.1: Facebook asks you to solve a CAPTCHA before it serves the actual response

For example, Facebook does not want to be scraped automatically, so without a valid user session it asks you to solve a CAPTCHA before you are shown the information you are looking for. Also Google analyzes the requests from users and when the behaviour becomes suspicious, it asks the user to enter the given text from a CAPTCHA.

Wrong response

Websites can detect in various ways whether a request comes from a web scraping software or a real users. There are websites that return different responses to web scrapers than they do to real users using a browser.

The professional network LinkedIn, for example, checks the request's User Agent and tries to find out, whether this could be an authentic web browser or not. If that is not the case, they reject requests with a custom HTTP error code 999 (*Request denied*) without serving the actual information in the response body. The HTTP status code 999 is not specified in the corresponding RFC [13], so returning with this status code might indicate that scraping their website is strongly unwanted. Doing the same request with a valid browser's

LISTING 3.1: Hidden strings in personal data

```
<span class="hide">rkn</span>P<span class="hide">nlig</span>
arallelst<span class="hide">7cd</span>r<span class="hide">ayi
</span>.<span class="hide"> 82 </span>4<span class="hide">
7</span>1,<span class="hide"> 01</span>4<span class="hide">
1</span>8<span class="hide"> </span>6<span class="hide">
76</span>8<span class="hide">
</span>3A<span class="hide">
r</span>ha<span class="hide">hjr</span>u<span class="hide">
cn</span>s
```

User Agent string serves the correct response including a *200 OK* HTTP status code. Next to that, according to user reports, LinkedIn seems to block specific IP ranges belonging to data centers, which usually contain headless server environments [26, 27].

Authentification

Many APIs and websites in general want users to authentificate before accessing them. Especially for social network sites, a user account is needed in most cases before a user may see other user's profile information. To use APIs, often a user has to authenticate on a webpage.

Content mixed with waste

Some sites try to prevent from web scraping by adding waste information between the correct information. An example from the German phonebook website DasTelefonbuch.de can be found in Listing 3.1. There are hidden strings in a phonebook entry which are just not shown in the client. In this case they just get hidden by using Cascading Style Sheets and can be filtered easily when the scraper is specifically adapted for DasTelefonbuch.de, but for automated scraper, this can hinder from finding the correct address.

Overview

In Table 3.2 you can see, what issues apply to what data souces. The selection is made based on popular services that might provide the needed information.

DasOertliche.de and *DasTelefonbuch.de* are websites that allow users to search in the official German phone book. Phone books are directories that enable people to find the phone number of other people. While those physical books played an important role in the past, today - next to the actual books - the phone book information is made available in the form of web applications.

Facebook is, with more than one billian daily active users worldwide, the biggest social network. Facebook users usually give public access to personal and even identifying data. 61% of facebook profiles show images that are suitable for direct identification, while 80% if all images contain at least some information useful for identification [16]. 88.8% disclose both their full birthdate (day and year) and gender on their profiles. The current place of residence is publicly available at 45.8% of the profiles. These statistics are about several thousands of students at Carnegie Mellon University from 2005 [16].

Twitter is a social network that does not store and show much information about a person, since it is more about the content, users produce than about the information about them. Nevertheless, many users use their real name as the shown name next to their twitter name. Using the official Twitter API, you are able to search for those names, but the only useful piece of information you can retrieve is the location. This can be used as a part of theverification process of a user's address. Eventually, the profile picture could also be used.

LinkedIn is a social network for professionals. There are APIs to access the data, but unfortunately, the people search part is not accessable by default. To gain access, one has to apply for LinkedIn's approval and agree on the rules for it. The rules for instance include, that it is only allowed to call the API within an active user session, which is not the case in our project.

TABLE 3.2: Some possible data sources and existent issues

Name	**robots.txt**	**CAPTCHA**	**Wrong response**	**Authentification**	**Waste**
DasOertliche.de					
DasTelefonbuch.de					(✘)
Facebook	✘	✘		✘	
Twitter	✘			(✘)	
LinkedIn	✘		✘	✘	
XING	(✘)			✘	
Google+	(✘)			✘	
StayFriends				(✘)	

3.2.3 Open web search

Many third-party data sources have a quite limited value for validating personal information of the users. Looking at the popular social network Twitter for example, there is only a name, place and a free text description. The name field is often used for pseudonyms and the free text description rarely contains any data that should be validated. Furthermore, it might be the case that users do not have any profiles on one of the static data sources that gets checked.

An alternative to such static data sources that get requested directly, it is possible to request a search engine whose main objective it is to crawl the web and create a as complete index of it as possible. Using such a search engine enables us to find occurences of the user in the whole web.

The major search engines Google, Bing, Yahoo, DuckDuckGo, Ask, and AOL disallow scraping the result pages in their robots.txt file. Lycos does not explicitly disallow scraping, but depending on the IP address the request comes from, the result page is empty. That is the case for requests coming from the University of Twente network. Whether this is a ban or any other limitation is not known. The URLs of the results point to an internal redirect page provided by Lycos which is not allowed to scrape according to the robots.txt. However, this internal redirect page contains a URL parameter containing the correct URL, so it can be extracted without breaking the robots.txt rules.

TABLE 3.3: Major search engines and their scraping policies for result pages

Search Engine	Allows scraping	Disallows scraping
Google		✘
Bing		✘
Yahoo!		✘
DuckDuckGo		✘
Lycos	✘	
Ask.com		✘
AOL		✘

Search term composition

To find a well working approach for the open web search, we will compare different parts of the search. At first, we compare the results of different search approaches to find out whether the given results are actually relevant. For us a search approach is the composition of a search term string. For instance, one approach can be searching for the full name, another can be searching the address or a combination of both.

Then, for the open web search, there is one big challenge. We neither know what actual data source is used, nor how the data source presents it's data. When we want to search for an address on a random webpage, this is an easy task for a human being. But for a machine, this is way more difficult. In the early 2000s, microformats and a sementic web in general was proposed and standards introduced to make such information machine-readable in web pages [20, 1, 18]. Today, it is still not really common to use these standards, so it would not be helpful to rely on them. They still can support us in finding the wanted information.

Information extraction

Once a potential data source is found, we still do not know how the information is structured on the specific data source. In contrast to a previously known data source, the open web search gives us results that we probably have never seen before.

In order to find a way to find the information we need, we compare different methods to extract information from the found web pages. This includes, dependend on the information type, regular expression or lexicon based approaches.

3.3 Modes

There are various approaches to validate user information. In this thesis, we differentiate between the modes *ByPrimaryInformation*, *BySecondaryInformation* and *ByMixedInformation*.

The **ByPrimaryInformation** mode follows the approach that information items are validated by seaching for exactly this information item, while the BySecondaryInformation mode follows the approach to search for an information item in order to validate another information item. In ByPrimaryInformation mode for instance, the validator searches for the name of a person and checking information like the address and the phone number to conclude the authenticity of the name.

In the mode **BySecondaryInformation** it is exactly the other way round, so when verifying the name, it searches for both the address and the phone number and then compares the actual found names with the name of the person to be verified. Table 3.4 visualizes, what information is used in both modes.

ByMixedInformation works in a slighly different way. In this mode, all information in combination plays a role. In data sources where the data structure is known in advance, information belonging together to one person can be identified. When doing so, the most similar person can be calculated and the information belonging to this most similar person is compared with the person to be verified.

TABLE 3.4: Information that is used for verification

ByPrimaryInformation

to be verified	*search for*			*check/compare*		
	name	address	phone	name	address	phone
name	✘				✘	✘
address		✘		✘		✘
phone			✘	✘	✘	

BySecondaryInformation

to be verified	*search for*			*check/compare*		
	name	address	phone	name	address	phone
name		✘	✘	✘		
address	✘		✘		✘	
phone	✘	✘				✘

The identification is done by comparing all person information items while each information type is weighted differently.

$$sim_{overall} = \frac{3 * sim_{name} + 2 * sim_{address} + sim_{phone}}{6}$$

The overall similarity ($sim_{overall}$) is just used for comparison purposes to find the entry having the highest similarity with the person to be validated. The similarity values of the different information items (sim_{name}, $sim_{address}$ and sim_{phone}) are values between 0 and 1 like they are used in the other modes as well. A similar name has with the weighting factor 3 the highest influence on the overall similarity, followed by the address (weighting factor 2) and the phone numbers (weighting factor 1). The sum of those similarities is normalized to a value between 0 and 1 by dividing by 6.

To find the persons, it is searched for all information items seperately and the most similar person of the union set of search results will be used as the most similar person. Then, the authenticity will be calculated using the information belonging to this person.

The authenticity in this mode is based on the similarity as well. The similarity itself then is the second variable. The DasOertlicheValidator can be instantiated using one of five similarity measures as a paramenter: Jaccard, Jaro, Jaro-Winkler, Levenshtein and Damerau-Levenshtein. In Chapter 2.2, these string similarity measures are explained in more detail.

3.4 Design set and test set

For classification in machine learning, an existing data set is usually divided into seperate sets, the training set, valiation set and test set. The training set is used to give examples to the artificial intelligence (AI), so that it can learn from them, while the test set is used to let

the AI classify the data itself without knowing it before to be able to evaluate it.

The validation set is used in machine learning projects in order to prevent overfitting. Sometimes the training set and validation set are combined to one set, which is called design set [25] in reference to the fact that is is used during the whole design process.

In our case, machine learning does not apply in the classification process, but we still make use of the well-proven approach of separated sets. There is a total set of 100 records containing personal data. A subset of 42 records contains correct data of various employees of Tobit Software. The remaining records also contain information of employees, but they have at least one wrong piece of information in the records. Each information item is annotated as correct or incorrect.

This set is then splitted into a design set and a test set. During development only the design set is used for internal tests. The test set is used to actually run the actual experiment in order to evaluate the different approaches. The reason, why this is done is to prevent implementing a classifier that is too tightly coupled with the test set.

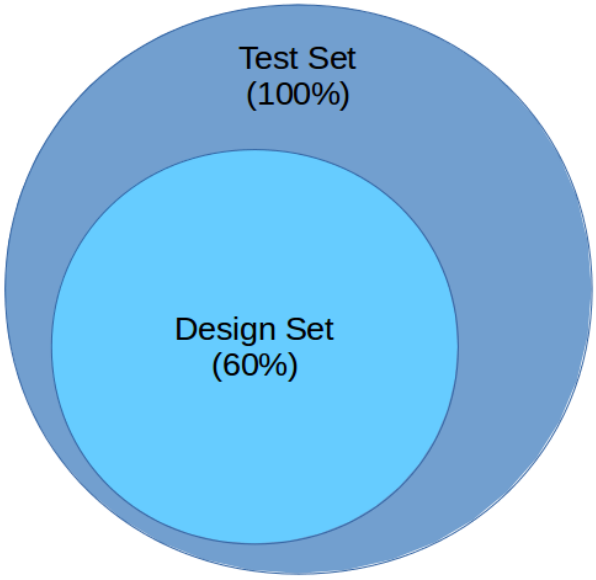


FIGURE 3.2: Division of the complete set into design set and test set

The design set contains 60% of the complete set, so that 40% of the set is hold back during development and is only used afterwards to evaluate the system with data that is in fact unknown until then. The actual separation is done by random selection.

3.4.1 The selection

The selection of data for the design set and test set needs to be reliable in terms of correct data. When evaluating a system that tries to find

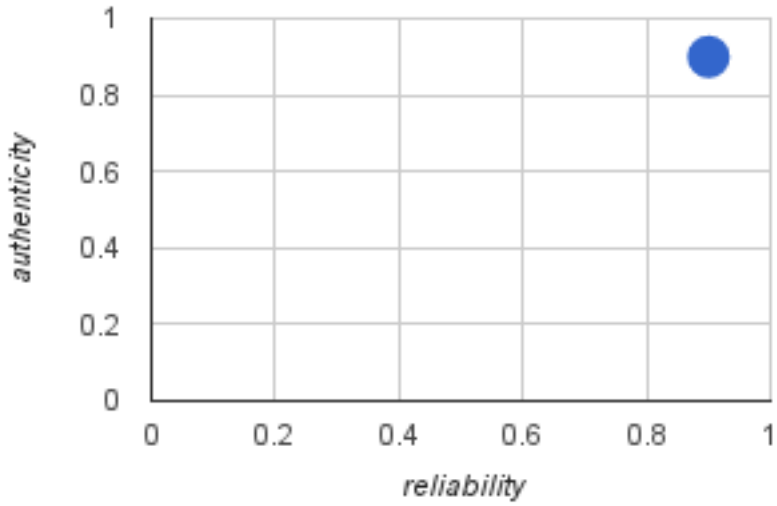
out whether information is correct or not, we need data that is known to be correct to check how the system performs.

Since we execute this experiment in the environment of Tobit Software, the complete set is a subset of their person database containing every kind of person the company deals with. As actual subset, 40 Tobit Software employees are randomly selected.

This is a quite limited selection of potential users. A broader selection of persons could of course lead to more significant results, but in order to evaluate that a classifier works properly, guaranteed correctly classified data is needed and whereas Tobit Software's general person database is a large dataset containing personal data, this personal data is not verified. That will be the task of this project. And since the subset of the database containing employees are well-known for Tobit Software, the user data of them can be seen as a reliable source of correct data. Since Tobit Software has employees in many different in-house departments, there is a rather high variaty in groups of people, meaning there are not only software developers, but also marketing experts, craftsmen, people working in catering services and more.

3.5 Result measure

FIGURE 3.3: Visualization of an example score



The system that verifies the user's information shall give a result score as a result. The score should contain the chance of correctness (*authenticity*) of information, but also an indication about the degree

of reliance (*reliablility*). The more indications you have for the decision of authenticity (e.g. multiple sources claiming the same information), the higher the reliability is. When information about a person barely can be found, the reliablility of the authenticity value is very low. Both values shall be represented on a scale from 0 to 1, where 1 is the highest.

This results in a two-dimensional score which can be visualized like in Figure 3.3, where both the authenticity and reliability is pretty high (0.9). Such a value would respresent, that the verified information has a high chance to actually be true.

In order to define a reliability value, multiple data sources are needed in combination. Otherwise, no statement can be made about the presence at mutiple locations. The open web search also allows us to define a reliability, because in fact it searches automatically for multiple potential data sources.

Chapter 4

Ethical Aspects

In this thesis, we discuss approaches that deal with personal data. Everytime personal data is an essential part of a software system, especially in automated environments, one has to think about ethical issues and how to deal with them. This chapter gives a brief overview about possible ethical issues that play a role in this project. The issues are not completely solved and answers are not given for every issue, but it shows what is important to think about when putting such a system into practice.

4.1 Ethical issues

Most papers about retrieving personal data from web sites and/or social networks state that doing so is done by criminals or unethical companies only. This already gives an indication of for what purposes these techniques are used usually. Then, what are the purposes to apply them in the context of this project and is it acceptable from an ethical point of view at all? Asking with other words, what would distinguish Tobit Software from criminals, if it does?

We have to start by looking at what criminals want. They, for instance, want to retrieve as much personal data as possible to send them spam, sell this data or even use it for identity fraud. These are all motivations that in some point harms the people whose data is gathered. This is definitely not the case for this project.

Nevertheless, personal data is always a sensitive topic. Companies should only collect, store and process data of their customers or business partnes, they gave the company theirselves knowing what the company uses it for. In our case, people fill in personal data and the system should find proof (or indication) for the correctness or incorrectness of this data somewhere at third-party online sources. But what should happen when the system decides, that the chance for the data being correct is very low and it maybe even finds the correct information somewhere else? Is it ethically acceptable to store the correct data additionally to the originally given data or even replace it automatically? In this case, you collect and store data, the person did not actively agree on. Therefore, users should be informed about

this verification process. At least the terms and conditions of the service that uses the verification system should contain a notice about the fact that information is double-checked using other sources on the internet.

Assuming the data which is more likely to be correct gets stored, but actually belongs to another person, and is showed up in the user interface. Then the system serves personal data of a random person found on the internet with the one who uses the system. This is not desirable.

Another aspect is the fact that one always has to reveal at least some pieces of personal data when looking for more data of that person. You have to send websites or any other kind of API which ideally contain personally identifiable information. So you serve this data to the API provider. For instance, you want to check the phone number simply by checking the phone book, then you usually send them the name and city of the person. In case of a phone book, the service is especially made for such requests, but first, the user probably did - again - not agree with giving the data to third parties, and second, there are other services that are not made for these purposes and you still send them the data. This should always be in mind when designing verification methods using third-party online sources.

When a request is made via the unsecure HTTP protocol, there is always the possibility for others to easily eavesdrop the communication. In this manner, sending personal data to trustworthy third-party online services can still become a data leaking issue.

4.2 Public data

From ethical point of view, is important, that the information used in the process of verification, is publicly available. Publicly available in this case means that is is public by purpose. On the internet, there are many resources that contain illegally gathered data. Sites like *pastebin.com* for example, are often used by hackers to publish confidential data from their victims. In an open web search, these sites could be found as well, but in such a case, the person probably did not agree on publishing the data, or even worse, they probably did not even know about it.

Next to that, publicly availabe means that the information can be retrieved by everyone and not only by a limited group of people. The condition of an obligate user account can still let information be publicly available in case everyone is able to create such an account without any further conditions or user rights to be granted.

In an open web search, we don't have influence on the results the search engine returns, so in such a case we never know, whether information is publicly availabe by purpose or not. To solve this issue, there are various approaches you can apply. What approach finally

should be used has to be weighted up against each other individually.

Whitelisting Every potential data source that should be allowed has to be defined in advance. This gives the best control about the data sources, but is also the least flexible. It would destruct the biggest advantage of an open web search, that is to be completely open for any data source without having them predefined.

Blacklisting All potential data sources that are not listed on this blacklist, are considered as data sources that can be used within the verification process. Data sources that are known to publish information that is not public by purpose should be blacklisted in this case. This approach can not cover all undesired data sources, but it could form a good trade-off between banning undesired data sources and the flexibility of an open web search.

Greylisting This approach is anything between whitelisting and blacklisting. You could think about a system that has both a whitelist and a blacklist, and when it comes across a new unknown data source it asks for a manual review by any real person who has to decide whether the data source has to be set on the blacklist or the whitelist. This approach has a very low chance to use data that is not public by purpose, but it loses the ability to do verification automatically because of the manual review-process.

Chapter 5

Experiment setup

While Chapter 3 showed a general overview about how a system, that verifies personal data, could look like, this chapter describes what we do in our experiment in order to answer the research question.

5.1 Method

To find a well-working approach that can verify persons' information using third-party webbased data sources, we do a series of experiments. Therefore, a prototype will be developed that enables us to vary specific parameters that represent different approaches, in order to compare them with each other.

There are two general types of using data sources; a predefined **specific data source** and an **open web search** without predefined data sources. As described earlier, the open web search then is used to find potential third-party data sources at public search engines. In our experiments, we use both, one specific data source (DasÖrtliche.de) and an open web search approach using the Lycos.com search engine.

Whereas the specific data source can use the information straightforwardly because the website's structure is already known, the open web search needs a generic way to extract information. Therefore, the open web search part is split into multiple subexperiments.

Search Term Composition Firstly, we want to find out what search term can be best used to find information about a specific person at the used search engine.

Information Extraction Then, we want to find out what approach performs best to extract specific information items from web sites whose structure is unknown.

OpenWebSearchValidator Finally, we use the findings of the first two subexperiments to design an actual validator.

5.1.1 DasOertliche.de

The DasOertlicheValidator is a validator using information from DasOertliche.de which is based on Germany's official phone book data. To access the data, requests are sent to the server and the resulting web site is scraped in order to extract usable information from it.

There are some variations for the DasOertlicheValidator that are compared in this experiment. One out of three different modes can be given as parameter for the validator. These three modes are *ByPrimaryInformation*, *BySecondaryInformation* and *ByMixedInformation*.

The ByPrimaryInformation mode follows the approach that information items are validated by seaching for exactly this information item, while the BySecondaryInformation mode follows the approach to search for an information item in order to validate another information item. In ByPrimaryInformation mode for instance, the validator searches for the name of a person and checking information like the address and the phone number to conclude the authenticity of the name, while in BySecondaryInformation mode, the validator searches for the address and the phone number and check for the presence of the right name in order conclude the authenticity of the name. The authenticity is then determined by the similarity of the found information compared to the stored information.

The third mode, ByMixedInformation, tries to find the most similar entry from the resultset of DasOertliche.de and then compare the informtation of this entry with the person to be validated.

The resulting authenticity is calculated by the most similar result of the searched information. If the address should be used for verification, the highest similarity between the found addresses and the the given address is used to represent the authenticity. The different similarity measures *Jaccard*, *Jaro*, *Jaro-Winkler*, *Levenshtein* and *Damerau-Levenshtein* are compared for this purpose and thus vary during the experiment

FIGURE 5.1: Information that can be extracted from DasOertliche.de

Figure 5.1 shows an example entry of DasOertliche.de containning name, address and phone number of a fictional person. This information can be extracted from the site by web scraping.

5.1.2 Open web search

Like explained in Chapter 3.2.3, there is only one search engine that allows us to scrape their result pages according to their robots.txt; Lycos.com. Hence, we will focus on Lycos for the open web search. The open web search is subdivided into the subexperiments of finding the best *search term* to find perons on the internet, how to *extract address information* from unknown web pages and the actual validator.

5.1.3 Search term composition

In this experiment we want to find out, how we can find result pages about a specific person using a search engine. Therefore, we check the following search approaches in the form of search string compositions:

- <first name> <last name>
- <first name> <last name> <city>
- <street> <city>
- <email>
- <phone number>

The results from the first result page are then retrieved for all of these search string compositions and manually annotated as relevant or not. A result is relevant, when the web page actually contains information about the person we are searching for, be it additional information or just the information we searched for anyway.

Having information about which result is relevant and which one is not, we can calculate the precision for comparison purposes.

5.1.4 Information extraction in open web search

Then the different approaches for information extraction are compared. In this specific case, we focus on finding postal address information that should be extracted. In general there are two different approaches that are compared in this experiment: **Regular expressions** and **lexicon based extraction** which is based on the approach Can et al. showed in their paper [5].

Regular expressions

The regular expression for address pattern matching has the following requirements[1]:

[1] These requirements are partly based on the official rules from the German Duden (http://www.duden.de/sprachwissen/rechtschreibregeln/strassennamen).

- It shall match a full address, meaning street name, house number, ZIP code and city after each other.
- The single parts of the address are seperated by arbitrary whitespaces.
- Between the house number and the ZIP code, there may also be a demiliter like "," additionally or instead of a whitespace.
- Between the ZIP code and city, there may be a comma as a delimiter as well.
- Street name
  - The street name has to begin with a capital letter.
  - Beginning from the second character, the street name may contain all letters and numbers, but also periods, dashes and spaces.
  - The street name's length is between 1 and 50 characters.
- House number
  - The house number can be a simple number.
  - The house number can be a simple number followed by an upper or lower case character (e.g. "7A". The character and the number can be seperated by a whitespace.
  - The house number can be a range of house numbers, represented by two numbers seperated by a dash.
- ZIP code
  - The ZIP code consists of a sequence of 5 digits.
  - The ZIP code may start with a country prefix ("D-") immediately before the actual ZIP code.
- City
  - The city may consist of a arbitrary sequence of characters.
  - The city may also contain dashes.

Listing 5.1 shows a regular expression that fulfills the requirements stated above. It is used for our experiment when extracting addresses by regular expressions.

LISTING 5.1: Regular Expression for address extraction

```
([A-ZÄÖÜ][A-Za-zÖÄÜöäüß\-\␣\.0-9]{0,49}\s
    [0-9]+(\-[0-9])?(\s?[A-Za-z])?[\s,\-|;]+(D-)?[0-9]{5}[\s
    ,]+[A-Za-zÖÄÜöäüß\-]+)
```

Lexicon based extraction

Can et al. [5] propose a way of extracting information from web pages by using prepared lexicons. In detail, they do this by following these steps:

1. Text segmentation
2. Tokenization
3. Type classification of tokens
4. Parsing by parsing rules

In their paper, they do text segmentation based on HTML elements and the visual appearance in order to find out what text parts belong to each other. We assume, that by now, webpages are often way more complexly structured than in 2005, when the paper is written. Therefore, we decided to use the text in a more flexible way and consider every text part of the website as potentially relevant. When somewhere in the whole text a pattern for an address can be found, this is still okay.

We begin by tokenizing the website's whole text parts. Each word is seen as a single token, each delimiter such as "," or ";" is also split into seperate tokens. The text string "My address is: Mittelstraße 2, 44875 Bochum" would result in the following single tokens: "My", "address", "is", ":", "Mittelstraße", "2", ",", "44875" and "Bochum".

Then, we try to classify the types of the tokens. This is done by looking the terms up in prepared lexicons. Therefore, we create lexicons that include, for instance, all German city names or common street suffixes. So we try to find the token text in one of the lexicons, if it can be found, it can be classified as the type, the lexicon stands for.

city The city lexicon contains all German city names. The data is originated from the Federal Statistical Office of Germany using the data of European Union census in 2011.

street The lexicon contains many usual suffixes of German street names, for example "straße", "str.", "weg", "platz" or "allee".

zip The zip lexicon contains all German ZIP codes.

delimiter The delimiter lexicon contains a list of various delimiters such as comma, semicolon, dash or the vertical bar.

After that, we try to find specific *token compositions* in the list of tokens. In other words, we try to find specific predefined sequences of token types in the document in order to identify the belonging information types.

To identify a postal address, for example the following token type sequences would match:

- <street> <number> <zip> <city>
- <street> <number> <delimiter> <zip> <city>
- <street> <housenumber> <zip> <city>
- <street> <housenumber> <delimiter> <zip> <city>

The difference between a number and a house number is that a number only matches pure numbers while a house number would match house numbers like "7-9" or "9 A" as well. Both types of numbers can occur in addresses.

5.1.5 Open Web Search Validator

Using the findings from the search term and information, we will create an OpenWebSearchValidator for the ValidationService. The OpenWebSearchValidator has the following parameters that can change its behaviour.

Search Engine The search engine that should be used to find potential data sources (e.g. *LycosSearch*).

Search Term The search term composition that should be used in for the search for potential data sources in the given search engine (e.g. *fullname*, *address*, *fullname+address*).

Information Extractor The information extractor that should be used to extract the desired information from the data source (e.g. *RegexExtractor* or *LexiconExtractor*).

Similarity Measure The similarity measure that should be used for comparison between actual and found information (e.g. *Jaccard*, *Jaro*, *Jaro-Winkler*, *Levenshtein*, *Damerau-Levenshtein*).

In our experiment, we will use the Lycos websearch as the only search engine and the best performing information extractor, based on the findings of the belonging experiments. Also the search term composition will be chosen based on which one returns the best results.

5.1.6 Baseline

Next to the actual validators, there is an additional validator called *RandomValidator*, which decides randomly whether an information item is authentic or not. Both authenticity and reliability are determined as each a random number between 0 and 1. This validator is used for comarison purposes. An actual validator should have a better performance than this baseline implementation.

5.2 Technology

The prototype that will be developed shall be built in a way that it can be used for this experiment series, but also as generic as possible, so that Tobit Software as practical case of this problem would be able to reuse the concepts and parts of the code respectively the whole system in general easily in case they want to implement a solution as it is used in this thesis.

5.2.1 Software Requirements

The prototype to be developed has as main goal to evaluate different approaches to verify the authenticity of personal data. Another goal is to make it easy to convert the prototype into a software system for usage in a production environment.

The following list of requirements is prioritized according to the MoSCoW prioritisation approach. MoSCoW puts requirements into four different priority categories; "**M**ust Have", "**S**hould Have", "**C**ould Have" and "**W**on't Have" [23].

In Table 5.1, you can find an overview about the software-side requirement specification follwing the MoSCoW approach.

TABLE 5.1: Requirements for prototype

Req-ID	Description	Priority
R01	The system shall receive personal data as input.	Must
R02	A personal data shall consist of the following information items: first name, last name, street, zip, city, email, phone, birthday	Must
R03	The system shall return a score containing authenticity (0-1) and reliability (0-1) for each information item.	Must
R04	The system shall be able to validate by using multiple data sources.	Must
R05	The system's data sources should be extendable as seperate modules.	Should
R06	The system's data sources should be able to be executed with different parameters to compare different approaches.	Should
R07	The system could serve an API (e.g. RESTful) that can be used for sending validation requests to the system and returning the belonging response.	Could
R08	The system will not store other personal data than the data which is already available for a person.	Won't

5.3 Architecture

The prototype will be developed in Python. Python is an interpreted programming language, whose interpreter is available for multiple platforms. It is used for a lot of web scraping projects, thus, there are a lot of stable standard procedures and libraries, many people rely on. In general, it provides reliable and stable technologies for web scraping purposes. This makes it fit for this project.

To archieve the modularity of the prototype, an object oriented design pattern is chosen for implementation. There are the classes Person and EvaluationPerson. An instance of Person contains information such as name, address, phone number, e-mail address and the date of birth, while the EvaluationPerson is a specialization of a Person adding fields that indicate whether the information items are correct or not. This construct forms the data structure that holds the data which forms the test set.

In a productive environment, the Person class would be used to create objects that get passed as input for the process. In our experimentation environment, the inherited EvaluationPerson class plays an important role. Using this class we can evaluate the system easily, because we directly know, what information actually is correct or incorrect.

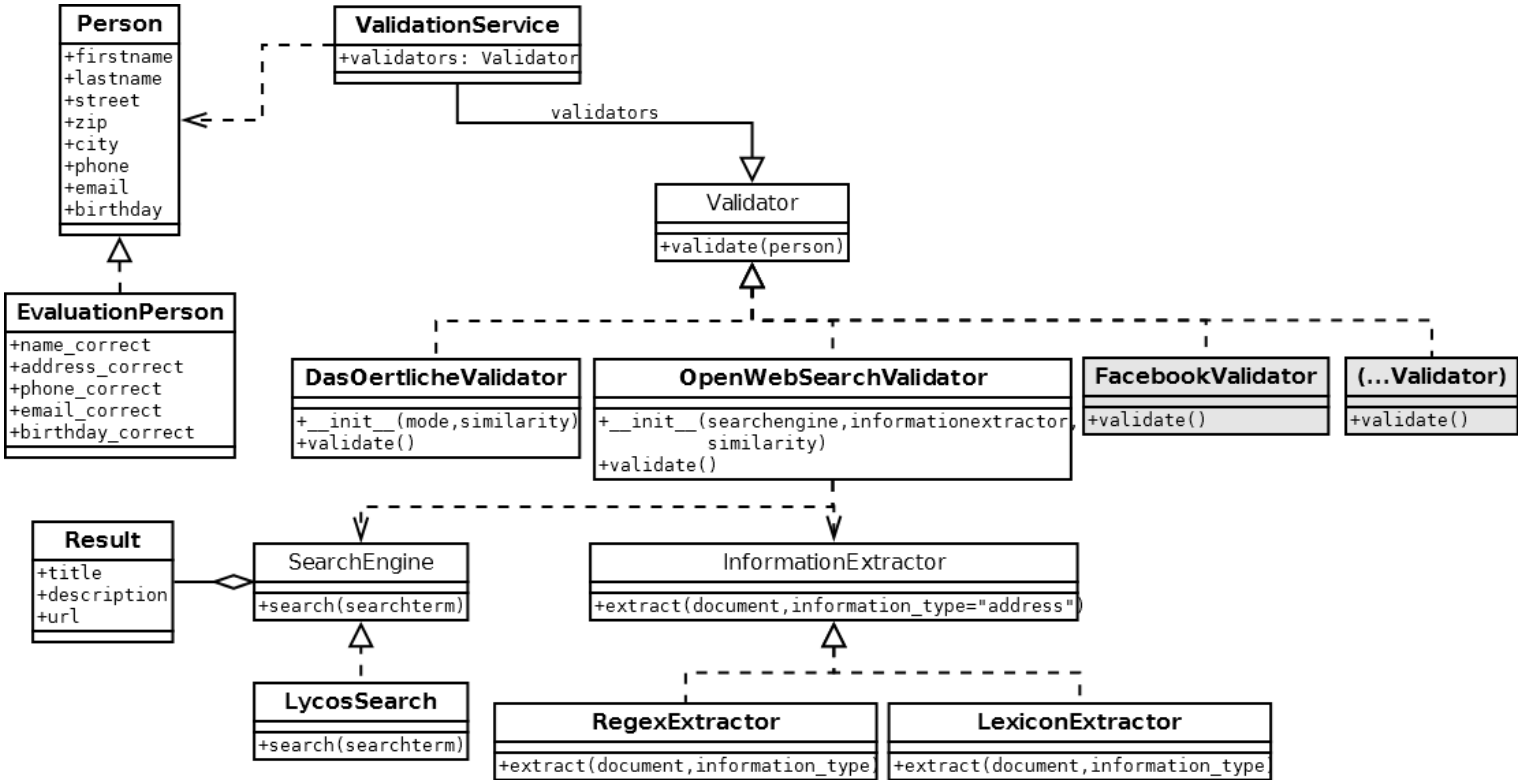


FIGURE 5.2: UML class diagram of prototype

The ValidationService is the central part of the prototype which actually starts the validation process. It holds multiple instances of Validator implementations (for different data sources) that should be used in the validation process. The Validator class itself is abstract, so that it only is used to form an interface that forces the children to implement the *validate* function.

Figure 5.2 shows a UML class diagram about the overall structure.

The DasOertliche validator, which makes use of DasÖrtliche.de as data source receives a mode and a similarity measure in the constructor, so that they can be chosen at the moment of initialization of this object.

The OpenWebSearchValidator itself is also an implementation of the abstract Validator class. It makes use of a search engine and an information extractor. The classes SearchEngine and InformatioxExtractor therefore are also absract in order to allow various different implementations.

The SearchEngine class, which is implemented by the LycosSearch class in our case, in able to return a list of Result instances, according to what the search result contains. Each result consists of a title, a description text (the text snippet you can see on the search engine's result page) and the URL. The URL from the LycosSearch is extracted from a custom result link the search engine sends the user to in order to redirect him or her to the actual web page. The implementation of LycosSearch skips this redirection step.

Moreover, the OpenWebSearchValidator makes use of an InformationExtractor. The class itself is abstract, so that, again, multiple different implementations are possible. In our experiment setup, we implemented a regular expression based information extractor (RegexExtractor) and a lexicon based extractor (LexiconExtractor).

In order to instantiate an OpenWebSearchValidator, one need to pass instances of implemented SearchEngine and InformationExtractor classes directly. The grayed out classes in the diagram are just for illustration purposes and will not be implemented. They show what theoretically would be possible to extend.

The whole architecture contains a lot of abstract classes. This makes the code modular and parts of them can easlily be changed without touching parts it does not directly deals with.

5.4 Evaluation

How we evaluate the results depends on the type of experiment. Our experiment is split up into several subexperiments; verification with a specific data source, how to find persons in an open web search, how to extract information from unknown websites and the verification using the open web search approach.

5.4.1 Verification

Specific data source

To be able to evaluate whether the prototype performs well or not, it is important to know what properties a well working prototype characterizes. In this case, there is no need for only correctly classified positives or correctly classified negatives, but about correct classifications in general, so true positives and true negatives in relation to the total amount of classifications are important. This *accuracy* called measure is only applicable for binary classifiers, so the non-binary authenticity from the score cannot directly be used to check the accuracy. Therefore, the results can be binarized by treating values lower than 0.5 as 0 and values higher or equal to 0.5 as 1.

The following formula is used to calculate the accuracy, while TP is the set of true positives, TN is the set of true negatives. FP and FN stand for the sets of false positives and false negatives.

$$accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|}$$

This is how we will evaluate the specific data source DasÖrtliche.de in this experiment, since the reliablility does not say anything in this environment without mutiple finds.

Open web search

As said, the accuracy does not take into account that the result score contains a reliability. To do so, it would be possible to only evaluate results having a reliability of 0.5 or above. An alternative is to check, how well the prototype perfoms using a custom evaluation value that takes all this into account. We specify the requirements for such a custom evaluation value as follows:

- The higher the reliability, the higher the impact on wrong authenticity decisions.
- If the information is actually correct, an authenticity value lower than 0.5 is bad, a value higher than 0.5 is good. A value of 0.5 is neutral.
- For incorrect information it is the same vice versa.
- A higher reliability value in combination with a bad authenticity value results in an even worse evaluation result and vice versa.
- The evaluation result is also a single value between 0 and 1 where 1 is the best value.

Having these requirements in mind, the following formula gives the desired results, where s_a is the authenticity of the score and s_r the reliability, while a is the actual authenticity (1=correct, 0=incorrect).

$$eval = 1 - ((|a - s_a| - \frac{1}{2}) \cdot s_r) + \frac{1}{2})$$

Using the results of this evaluation formula, implementations are comparable in a way that also takes the realiability into account. In our experiment, this is used for the OpenWebSearchValidator. In future, it could also be the base for an evaluation of multiple specific data sources.

5.4.2 Search term compositions

In our subexperiment about how to find persons on the web, we will look at the precision of the results. Therefore, all returned results have to be reviewed and annotated as relevant or not manually. Relevant in this case means, that a result points to a web resource that is about the actually searched person.

$$precision = \frac{|results_{relevant}|}{|results_{total}|}$$

For us, it is also interesting, how for how many persons, at least one relevant result could be found, because it for us it is more relevant to find information about as many people as possible instead of finding much information about just a few specific persons. Therefore, we evaluate the ratio of persons that returned at least one relevant result; $ratio_{relevantPerson}$ according to the following formula, where $P_{relevant}$ is the set of persons having at least one relevant result and P_{total} is the set of all persons.

$$ratio_{relevantPerson} = \frac{P_{relevant}}{P_{total}}$$

5.4.3 Information extraction

The information extraction subexperiment wants to find out, how to best extract (postal address) information from websites. Thus, relevant results are the amount of found address instances and how many of them are correct or respectively wrong. We then will make use of the common values *precision* and *recall*.

$$precision = \frac{|documents_{relevant} \cap documents_{retrieved}|}{|documents_{retrieved}|}$$

$$recall = \frac{|documents_{relevant} \cap documents_{retrieved}|}{|documents_{relevant}|}$$

For the final evaluatation, the F_1 score, a weighted average of precision and recall, is used.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

Chapter 6

Experiment Results

In this chapter you can find the results of the experiments. Firstly, it will give the results of a baseline implemenation with random results. Then, the results of the validator for DasÖrtliche.de are presented and discussed. Finally, the results of the open web search are shown, which are splitted into the different subparts of finding the best search term composition and how to extract information from web pages, followed by the results of the actual OpenWebSearchValidator.

6.1 Baseline

The RandomValidator, which is used as a baseline for comparisons is executed three times, since it can produce different results. In Table 6.1 you can see the accuracies of the results of all these three executions including the average.

TABLE 6.1: Accuracy for baseline (RandomValidator)

Information item	Iteration 1	Iteration 2	Iteration 3	Average
name	0.58	0.52	0.49	0.53
address	0.52	0.40	0.48	0.47
phone	0.51	0.51	0.53	0.52

As expected, the accuracy is about 0.5 for all information items. This means, that half of the randomly classified results are correct.

6.2 DasOertlicheValidator

Each run of the experiment stores the results of each validation including the authenticity, the reliability and the actual authenticity per individual information item.

In Table 6.2 you can see an overview about the accuracy of the resulting authenticity for the DasOertlicheValidator per information item type using the modes ByPrimaryInformation and BySecondaryInformation with each all the five different similarity measures.

TABLE 6.2: Accuracy for modes ByPrimaryInformation and BySecondaryInformation

ByPrimaryInformation			
Similarity-Measure	**name**	**address**	**phone**
Jacccard	0.52	0.42	0.15
Jaro	**0.83**	**0.63**	0.21
Jaro-Winkler	**0.83**	**0.63**	0.21
Levenshtein	0.67	0.45	0.15
Damerau-Levenshtein	0.67	0.45	0.49
Average	*0.70*	*0.52*	*0.24*

BySecondaryInformation			
Similarity-Measure	**name**	**address**	**phone**
Jacccard	0.45	0.60	0.50
Jaro	**0.65**	**0.70**	0.50
Jaro-Winkler	**0.65**	**0.70**	**0.56**
Levenshtein	0.47	0.65	0.54
Damerau-Levenshtein	0.47	0.65	0.54
Average	*0.54*	*0.66*	*0.53*

It is easy to spot the best result in the table; the validation of the name using the Jaro or Jaro-Winkler similarity measures. In Figure 6.1 you can see a number ray showing the distribution of the authenticities using the name validation with Jaro-Winkler limited to the actually correct names. Thus, each single point on the ray is an authenticity value the validator predicted for an actually correct name. As you can see, there are almost only results in the half above 0.5. There are only nine incorrectly classified names for actually correct names, which all received an authenticity of 0.

In general, there is a significant difference in the accuracy between the different information types. The name validation, for instance, generally scores better than the phone number validation, except for the case with mode BySecondaryInformation and Jaccard similarity measure. This is also the only case where the name validation scores worse than the baseline RandomValidator.

Another remarkable fact is that the validation of phone numbers, especially in the mode ByPrimaryInformation, is so significantly worse than the baseline RandomValidator. This phenomenon can be explained by how the phone validation works in ByPrimaryInformation mode: DasOertliche.de is requested with a search for the phone number and the result is then checked for the other information like name and address. This means, that the person to be validated has to state his or her phone number in the phone book. Since the persons in the person database, for instance, often have their personal mobile phone number or an own personal phone number instead of a

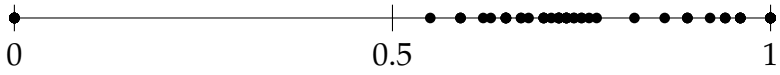


FIGURE 6.1: Distribution of authenticities for Jaro-Winkler

generic phone number for the whole familiy, there can't be found any entry, even if the phone number is actually correct. Another issue with searching for phone numbers is the fact that phone books are originally made for a search the other way round and this approach is a so called reverse phone book search. Such a reverse search has to be consented by the person who owns the phone number, which is often not done due to privacy concerns.

Would a phone validator with an accuracy of 0.15, that simply inverts the results before returning them, become a well working validator then? Irrespective of the fact that is then does not follow a logically comprehensible approach, the evaluation within the exaxt same data set would lead to a higher accuracy, indeed. But pushing the test set into extreme values shows that this approach could not solve the problem. Assuming, we have a situation having such an extreme test set with for example only correct numbers that are listed and searchable in the used phone book directory, it would give the exact opposite of the expected good value. The same would happen when doing this with the opposite extreme test set, only wrong numbers, that are not listed in the phone book directory. In our test set, 57% of the phone numbers are correct, but only very few numbers are listed in the used phone book directory.

TABLE 6.3: Matches in DasOertliche.de directory for ByMixedInformation

Similarity-Measure	***Exact match***		***Address match***		***Last name match***		***Phone match***	
	absolute	**relative**	**absolute**	**relative**	**absolute**	**relative**	**absolute**	**relative**
Jacccard	5	11.9%	19	45.2%	41	97.6%	2	4.8%
Jaro	5	11.9%	16	38.1%	41	97.6%	1	2.4%
Jaro-Winkler	5	11.9%	16	38.1%	41	97.6%	1	2.4%
Levenshtein	4	9.5%	14	33.3%	39	92.9%	0	0%
Dam.-Levensht.	4	9.5%	15	35.7%	39	92.9%	1	2.4%

TABLE 6.4: Information that is considered as matching

Inform. Type	Stored inform.	Found inform.
Name	Lukas Becker	Becker Lukas und Anna
Name	Linda Wagner	Schulze-Wagner Linda
Name	Monika Huber	Huber Michael u. Klein-Huber Monika
Address	Stadtstraße 15	Stadtstr. 15
Address	Berlinerstraße 2	Berliner Str. 2
Address	Einsteinweg 3	Einsteinweg 3a

The mode ByMixedInformation makes it possible to look at the single phone book entries that are found, since an identification of all the person's information takes place. In Table 6.3, there is an overview about how many of the found entries actually match exactly, but also how many found entries match in terms of the correct address, last name or phone number.

The matches are evaluated manually, so when the streetname in the database is stored as "Parallelstraße 41" and the found entry contains the abbreviated version "Parrallelstr. 41" it is considered as match, since the same address is meant. The same applies for example for double-barreled names that match the correct single name. Table 6.4 shows some examples of name or address constructions that is considered as matching even though they are not exactly the same. The names in this table are anonymized, but the composition of names or addresses are taken from the actual experiment.

In all cases where the address matches, the last name matches as well. About half of the results' addresses match with the addresses in the database, and even if there is no matching first name, this can be seen as a strong indicator for correct information. Often there live multiple persons (parents, children and probably some more) in one household and only one or at least just a few of them are listed in the phone book (only the parents).

As you can see, there is not much difference between the different similarity measures. Although the Jaccard similarity measure performs best in total, it is not significantly better than the other similarity measures.

Table 6.5 shows the accuracy of the mode ByMixedInformation for the different information types. The validation of the phone attribute in mixed mode are all around an accuracy of 0.45 and thus beneath the random baseline implementation. For address validation, the Jaro based similarity measures both perform best with an accuracy of 0.69. The same applies for the name validation, where the Jaro and Jaro-Winkler lead to an accuracy of 0.81.

TABLE 6.5: Accuracy in mode ByMixedInformation

ByMixedInformation			
Similarity-Measure	**name**	**address**	**phone**
Jacccard	0.77	0.57	0.46
Jaro	**0.81**	**0.69**	**0.45**
Jaro-Winkler	**0.81**	**0.69**	**0.45**
Levenshtein	0.80	0.60	0.44
Damerau-Levenshtein	0.80	0.60	**0.45**
Average	*0.80*	*0.63*	*0.45*

6.3 Open web search

6.3.1 Search results

For the search results, we requested the search engine of Lycos.com with the different search string compositions. Initially, it was planned to do a search on the email address and phone number as well, but this turned out to be unpractical for our case. The search requests including the phone number as search term lead to an empty search result. Why this is the case cannot be identified, probably Lycos.com simply does not want anybody to search for phone numbers and hence blocks these requests. In contrast, the email search terms are misleading since most email entries use an office email address from Tobit Software and Lycos.com does not search for the mail adress as a whole but seperates the parts in the email and searches for those. This combination leads to mostly non-relevant results that have to do with Tobit Software as a company.

For each search term and person, the first Lycos search result page is requested, so there are up to 10 search results each. Each search result is evaluated manually as relevant or not. In case of doubt, when there are no indications for being the right person, it is considered as being not relevant. This is often the case for Google+ profile pages which are not actively used where ony a first and a last name is visible and nothing else.

In Table 6.6, there is an overview about the precision of the first result page for a search operation at Lycos.com. It gives insight about how many results from the total result set is actually relevant. The second column shows the ratio of persons that returned at least one relevant result in on the first result page ($ratio_{relevant}$).

$scrpable_{relevant}$ in the thirds column shows the ratio of how many pages from the results are scrapable according to the robots.txt.

It is noticable that the precision that the $ratio_{relevantPerson}$ if way better than the overall $precision$ for the search term composition containing only the first and last name, while it is the other way round for the search term composition containing the city next to the first and last name. Why this is the case, gets clear when looking at the

TABLE 6.6: Precision of search approaches for Lycos.com search engine

Search term	$precision$	$ratio_{relevantPersonN}$	$scrapable_{relevant}$
[firstname] [lastname]	0.19	0.59	0.86
[firstname] [lastname] [city]	0.36	0.19	0.89
[street] [city]	0.06	0.26	0.96
[email]	N/A	N/A	N/A
[phone number]	N/A	N/A	N/A

different single results. It seems that the city name gives a quite strong emphasis on the city and the actual name gets less important for the search. Then Lycos.com tends to serve results about the city or sites about other persons who have the same last name in the given city. For example, family members are more often found when adding the city to the search term. Unfortunately, we cannot add concrete examples without losing the anonymization for this phenomenon.

Searching for the street address and the city doesn't seem to work very good. This search term composition gives too much stress on the area where the address is located. House numbers are ignored in most cases, so the results end up in being very generic; for instance street catalogues of a specific city or news articles about accidents in the given street.

Another finding is, that for common names it is more likely to give less relevant results. In one case, there is a to be verified person, whose last name is the name of a singer and the first name is the name of a song of this singer. Moreover, most persons tend to have either just a few relevant results (1-4) or almost all the results are relevant (7-9). For the search term composition <*firstname*> <*lastname*> there is no person that lead to 5 or 6 results. Though, the majority of persons (41.7% of all persons; 68% of persons having at least one relevant result) did not have more than two relevant results.

6.3.2 Information extraction

Table 6.7 shows the results of the information extraction experiment. The colums show the amount of total addresses that could theoretically be found on the checked websites, the amount of actually found addresses, how many **F**alse **N**egatives and **F**alse **P**ositives it had and also the **P**recision, **R**ecall and the balanced F-score of those results (**F1**).

It is easy to spot, that the regular expression based extraction scores significantly better than the lexicon based approach. The precision cannot be better, so none of the extracted addresses was no address. In total there were 17 results, that had minor mistakes in the result. For instance, a label like "Address" or a company name

TABLE 6.7: Results of address extraction

	Total	Found	FN	FP	P	R	F1
Regular Expression	237	226	11	0	1.0	0.95	0.98
Lexicon Based	237	28	209	0	1.0	0.12	0.21

stands in front of the actual adress. These results are considered as a correct address, since they all were limited to just small strings before it and the actual address is still in the result.

The false negatives, so addresses that were not retrieved by the regular expressions, are addresses that use other types of delimiters for seperating the street with house number and zip code with the city. Some results for examle used the capital letter "I" in order to show a vertical line as a delimiter in the address. Other websites use bullets that were not initially taken into account when creating the regular expression. Furthermore, sometimes addresses were mentioned in a sentence-like structure, where the word "in" can be seen as the delimiter. This is also not covered by the regular expression.

The lexicon based extractor did not do the minor mistakes, so here no company names or labels could be found in the results. This is the case because of the bit more strict rules this approach has naturally. Though, these strict rules lead to the worse score as well. The problematic aspect for this approach seems to be the strict list of street suffixes, where the street names are limited to in order to be found. This can be a problem for German street names and can probably work better for street names outside Germany. At least, the accuracy Can et al. reached with this approach was 89.3% for non-German addresses.

6.3.3 OpenWebSearchValidator

TABLE 6.8: Accuracy and eval-value for OpenWebSearchValidator

Similarity-Measure	Accuracy	Eval-Value
Jacccard	0.57	0.51
Jaro	0.70	0.55
Jaro-Winkler	0.70	0.55
Levenshtein	0.56	0.52
Damerau-Levenshtein	0.56	0.52

Like mentioned in Chapter 5.1.5, this experiment uses static OpenWebSearchValidator parameters for the search term and the information extractor based on their results. Since the information extraction with the regular expressions had the best results, the RegexExtractor

is used together with the search term composition using the combination of first and last name, which returned at leat one relevant result for 59% of the persons.

The results can be found in Table 6.8, where you can find the accuracy values for each similarity measure, but also the results of the eval value as described in Chapter 5.4. Obviously, the best performing similarity measure in the open web search environment are the Jaro measures with an accuracy of 0.7, whereas the eval value is 0.55, which is also the highest value for the open web search.

Chapter 7

Conclusion

This thesis answers the initial research question, whether information about a user or person can be verified automatically using publicly available third-party web resources. To be able to do so, we did a series of experiments that find an answer to the subquestions. In the following section we want to answer those subquestions in order to finally answer the research question.

Where can information of the users be found?

In this thesis, we investigated various data sources and the corresponding issues with those data sources or in general. There are data sources that hold lots of personal information, but in many cases, these data sources do not allow usage of the data for our purposes. In most cases the robots.txt disallows pages containing the personal information (e.g. Facebook, Twitter, LinkedIn, XING, Google+). While these rules are not a technical issue, their purpose is to tell web scrapers as ours that scraping these sites is unwanted.

Other issues, in many cases, can be bypassed technically as well, but since it violates the terms and conditions of the services, this should not be done in practice. From the checked data sources, only the phone book directory DasÖrtliche.de did not have any issues for scraping.

Next to that, an open web search is evaluated. That is to use publicly available search engines in order to find potential data sources, so that the actual data sources are not known previously. Here, we were able to find relevant websites for 59% of the persons using the Lycos.com web search.

In general, it is very dependant on how public the people to be verified are. How public people are, means how they deal themselves with their personal data. People who present themselves in a very open manner on internet profiles or personal websites are certainly easier to be found. This research has shown, that still 41% of the people cannot be found on the internet easily in an automatical way. But even people who disclose lots of information about themselves can be hard to find, when they, for instance, have a very common name.

What information should be searched for to find persons?

In the open web search validator, we checked various search term compositions. For 59% of the persons at least one relevant result could be found by using the first and last name as search term at the Lycos.com search engine. Even making the search term more specific by adding the person's home city, did not improve the results but dropped to 19%. So when searching for a specific person, using the full name without any additions is clearly the most successful approach.

In the validator for DasÖrtliche.de, the different modes represent what information is searched for. The ByPrimaryInformation mode, for example, uses the information to be validated as base for the search request. Thus, in this case, we cannot say, what information should be searched for in order to find a person's manifestation in this data source in general, but in can give an indication about what information should be searched for in order to verfify specific information items. Overall, it can be said that for name verification, searching for the name itself works best, since it gives the best accuracy value of 0.83. To verify an address, searching for the name and phone number in combination (BySecondaryInformation) turned out to be the best working approach. The same applies for the the phone number verification, but in this case, it is as important to state, that searching for the phone number (ByPrimaryInformation) cannot be recommended. The average accuracy is 0.24 for this approach.

How should the found information be compared with given information?

In each step, we compared different methods to compare the given information items with the found ones. Since the similarity forms the base for the result of the verification process, this plays an important role. The overall result is, that the Jaro-based similarity measures are the most viable ones. In fact, the Jaro and Jaro-Winkler measures had very few differences, but Jaro-Winkler is slightly better for this purpose, so this the one that should be used for such a system, for all information types, except the phone number where no similarity measure was used for.

Can information about a user/person be verified automatically using publicly available third-party web resources?

The initial research question asks, whether information about a person or user can be verified automatically using publicly available third-party web resources. This question can be answered positively to a certain extent. The verification is strongly dependent on the information a person makes available online.

Therefore, the selection of data sources plays an important role as well. A system that can do user information verification with external web resources needs to look search in resources, the person is present. For that reason, it is neccessary to think about data sources the potential users are present. A selection of data sources that covers the own user target group as best as possible. For example, in our test set, only 12% of the persons could be found with the same name and address at DasÖrtliche.de. This complicates the whole verification process.

Bibliography

[1] Ben Adida et al. “RDFa in XHTML: Syntax and processing”. In: *Recommendation, W3C* (2008).

[2] SCHUFA Holding AG. *Geschichte der SCHUFA*. 2015. URL: https://www.schufa.de/de/ueber-uns/unternehmen/geschichte-schufa/.

[3] SCHUFA Holding AG. *SCHUFA in Zahlen 2014*. 2014. URL: https://www.schufa.de/de/ueber-uns/unternehmen/schufa-zahlen/.

[4] Statistisches Bundesamt. *Bevölkerung - Verteilung der Einwohner in Deutschland nach Altersgruppen am 31. Dezember 2014*. 2015. URL: http://bit.ly/1ZbvIrR.

[5] Lin Can et al. “Postal address detection fromweb documents”. In: *Web Information Retrieval and Integration, 2005. WIRI'05. Proceedings. International Workshop on Challenges in*. IEEE. 2005, pp. 40–45.

[6] Peter Christen. “A comparison of personal name matching: Techniques and practical issues”. In: *Data Mining Workshops, 2006. ICDM Workshops 2006. Sixth IEEE International Conference on*. IEEE. 2006, pp. 290–294.

[7] Fred J Damerau. “A technique for computer detection and correction of spelling errors”. In: *Communications of the ACM* 7.3 (1964), pp. 171–176.

[8] Alex Schultz (Facebook). *Letter from Facebook's Alex Schultz*. 2015. URL: http://de.scribd.com/doc/287927116/Letter-from-Facebook-s-Alex-Schultz.

[9] Inc. Facebook. *Community Support FYI: Improving the Names Process on Facebook*. 2015. URL: https://newsroom.fb.com/news/2015/12/community-support-fyi-improving-the-names-process-on-facebook/.

[10] Inc. Facebook. *Facebook Q3 2015 Results*. 2015. URL: http://files.shareholder.com/downloads/AMDA-NJ5DZ/1179477464x0x859098/DC6C9112-AFF6-4E76-9168-7DBA0D5FFDAB/FB_Q3_15_Earnings_Slides_FINAL.pdf.

[11] Inc. Facebook. *Help: What names are allowed on Facebook?* 2015. URL: https://www.facebook.com/help/112146705538576.

[12] Inc. Facebook. *Help: What types of ID does Facebook accept?* 2015. URL: https://www.facebook.com/help/159096464162185.

[13] R. Fielding and J. Reschke. *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*. RFC 7231. RFC Editor, 2014. URL: http://www.rfc-editor.org/rfc/rfc7231.txt.

[14] Electronic Frontier Foundation. *Open Letter to Facebook About its Real Names Policy and*. 2015. URL: https://www.eff.org/document/open-letter-facebook-about-its-real-names-policy.

[15] Statista GmbH. *Bevölkerung - Verteilung der Einwohner in Deutschland nach Altersgruppen am 31. Dezember 2014*. 2015. URL: http://de.statista.com/statistik/daten/studie/382409/umfrage/verteilung-der-bevoelkerung-deutschlands-nach-altersgruppen/.

[16] Ralph Gross and Alessandro Acquisti. "Information revelation and privacy in online social networks". In: *Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. ACM. 2005, pp. 71–80.

[17] Artur Janc and Lukasz Olejnik. "Feasibility and real-world implications of web browser history detection". In: *Proceedings of W2SP* (2010).

[18] Rohit Khare and Tantek Çelik. "Microformats: a pragmatic path to the semantic web". In: *Proceedings of the 15th international conference on World Wide Web*. ACM. 2006, pp. 865–866.

[19] Vladimir I Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. 1966, pp. 707–710.

[20] Yuefeng Li and Ning Zhong. "Web mining model and its applications for information gathering". In: *Knowledge-Based Systems* 17.5 (2004), pp. 207–217.

[21] Facebook Ireland Limited. *Terms of Service*. 2015. URL: https://www.facebook.com/legal/terms.

[22] S. Madhu et al. *Risk assessment using social networking data*. US Patent App. 14/215,477. 2014. URL: https://www.google.com/patents/US20140282977.

[23] Iris Pinkster et al. *Successful test management: an integral approach*. Springer Science & Business Media, 2006.

[24] Joseph J Pollock and Antonio Zamora. "Automatic spelling correction in scientific and scholarly text". In: *Communications of the ACM* 27.4 (1984), pp. 358–368.

[25] Sarunas Raudys. *Statistical and Neural Classifiers: An integrated approach to design*. Springer Science & Business Media, 2012.

[26] Charltoons (Stackoverflow). *999 Error Code on HEAD request to LinkedIn*. 2014. URL: https://stackoverflow.com/questions/27231113/999-error-code-on-head-request-to-linkedin.

[27] Zoonman (Stackoverflow). *How to avoid "HTTP/1.1 999 Request denied" response from LinkedIn?* 2014. URL: https://stackoverflow.com/questions/27571419/how-to-avoid-http-1-1-999-request-denied-response-from-linkedin.

[28] Gilbert Wondracek et al. "A practical attack to de-anonymize social network users". In: *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE. 2010, pp. 223–238.

[29] William E Yancey. "Evaluating String Comparator Performance for Record Linkage". In: *Statistics* (2005), p. 05.