Bachelor’s Thesis

Degree Programme in Information Technology

NINFOS14

2018

Niklas Lindgren

RECOGNIZING NAMES OUT OF A STRING FIELD

– Sanitation of invoicing data

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Information Technology

2018 | 35

Lindgren, Niklas Tapani

RECOGNIZING NAMES OUT OF A STRING FIELD

- Sanitation of invoicing data

In Finnish e-invoicing standards, there is a lack of granular standardization which leads to information being contained in an arbitrary text field. It is important to parse these fields to their values and components in order to validate and utilize the data itself. The main goal of the work was to build a specific part of a data analysis system to parse arbitrary name strings and output good enough data for use in machine learning and big data. Results were production ready and highly accurate.

This thesis briefly introduces big data to explain the need for the technology discussed in this thesis and discusses the cause of the issues in the fields of debt collection and invoicing. Programming languages and operating systems are compared for the practical work. Name recognition uses many third-party data sources that are also introduced including dictionaries, name statistics as well as Application Programming Interfaces (APIs).

The logic for the name recognition is explained with multiple examples ranging from multiple persons to a mixed field with a business and a person. Name recognition itself uses technologies such as tokenization, classification, blacklisting and intelligent guessing to reach the highest possible accuracy with the data sets used.

The thesis concludes that name recognition logic built is sufficiently accurate for the needs of this use case and the results can be used to connect identities from different input strings. Even when a partial recognition happens, the output is normalized enough that results from two identical strings mostly lead to the same results even if one of them is minimally different.

KEYWORDS:

nlp, natural language processing, names, strings, tokenization, otc, order to cash

CONTENTS

FIGURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

Abbreviation	Explanation of abbreviation (Source)
DMV	Department of Motor Vehicles
API	Application Programming Interface
Kotus	Institute for the languages of Finland
BIS	Business Information System
OTC	Order to Cash

1 INTRODUCTION

The subject of this thesis is recognition of names, both persons, and businesses, out of a string field. The subject was chosen because of a need that arose for the company VISMA PPG Ltd. They have to process data that containing arbitrary text fields in their debt collection process. The focus in this thesis is the first line of the address field on a collection letter that contains the name of the receiver. This line can contain either a person’s name, business’ name or both. Incoherent and non-machine understandable data is one of the significant problems in machine learning and big data. The main goal of the work was to build a specific part of a data analysis system to parse arbitrary name strings and output good enough data for use in machine learning and big data. Perfect recognition was not the goal, but instead, best effort was the goal.

The thesis follows a structure where the commission is first explained in Chapter 2 and this explanation includes a high level overview of the problem solved by the thesis. In Chapter 3, there is a brief introduction to big data and the challenge in the industry that causes the need for this commision. In Chapter 5 the tools used for the project are compared and chosen, after which in Chapter 6 the recognition logic is explained with diagrams. Finally in Chapter 7, the conclusions based on the recognition logic and the suitability on the use-case are presented. Sources and references used are mostly technical documentation, publications of consults specializing in the area or straight from a specific source. There is a lack of academic sources, but the non-academic sources are deemed enough as the thesis is based on a practical approach to the problem.

2 THE COMMISSION

The commission was received from the company VISMA PPG Ltd. They are one of Turku's largest law firms, but they have a heavy business focus in Order to Cash (OTC) business, and it is a constantly growing business for them. Order to Cash refers to a business process that includes the whole life cycle of invoice handling from invoicing to debt collection. The commission of the thesis is a small part of the platform related to data sanitation, recognition, and analysis. This thesis focuses on recognizing a name out of a collection letter's name line by standardizing it and processing it. Because of historical reasons, there is little standardization for this field nor separation for first or last names as the field can contain data in multiple different formats. One of the causes is that the data is being supplied by the user and different users have different systems with different data models and formats. Sanitizing this data is important even just for legal reasons. The act on Preventing Money Laundering and Terrorist Financing (444/2017, Anti-Money Laundering Act) requires companies to identify their customers. In the scope of the name field, this is a considerable challenge as a name can be written in many ways (for example 'Niklas Lindgren' and 'Lindgren Niklas') and are thus considered as different names by traditional business logic. These issues grow when multiple data sources and differencing formats are taken into the equation.

3 BIG DATA

There are many definitions for big data depending on who is asked. This makes defining it in the scope of this thesis important. Here both of the terms ‘big data’ and ‘business data’ are explained in the context of this thesis to avoid misunderstanding.

3.1 What is Big Data?

Throughout this thesis, the term ‘data’ is divided into two sub-categories: the more traditional kind of data known as ‘business data’ and its newer cousin ‘big data’. Business data, which is sometimes also known as business-critical data, is used by businesses in their business applications for traditional business purposes. Good examples of business data would be names, phone numbers, and addresses in a customer information database or license plates and names of the cars’ owners in a Department of Motor Vehicles (DMV) database. Business data is what the businesses use to do business with their customers. All this business data is usually highly structured, well planned, well thought-out and all of it has a specific purpose: phone numbers can be used to call customers and addresses for invoicing purposes. In addition to these characteristics, the data is usually vital to the company’s survival and prospects.

Big data, on the other hand, is something that is either refined or derived from business data using different analysis methods or gathered as additional data when dealing with customers for example. If business data is defined as ‘structured’ and ‘planned’, big data is usually the opposite of that: it is large in volume, challenging to generate, capture, curate, search, and query. Big data is not without its challenges. These challenges include issues concerning what to gather, what is relevant and what is usable for further analysis and use. Legal issues include General Data Protection Regulation (GDPR) that requires a reason to store the data and sets other limitations such as how long it can be stored and how it can be used (GDPR, 2016). These issues may sound easy on the surface, but much planning and careful design are required as predicting future needs and uses for data can be are extremely be hard. The types of data that are possible to collect greatly depend on the service(s) the company provides to its customers and can be either extremely easy or hard to acquire. In the ideal cases, customers themselves provide nearly unlimited information about themselves for the exchange of the company’s goods or services. Good examples of these kinds of ideal scenarios are social networking platforms where users tend to share everything they can, even give the companies precious data, such as their phone contacts. When gathering data as a company, it is usually a good idea to store everything by default and later figure out if the data is useful.

3.2 Importance of Big Data

Companies traditionally have their core business revenue coming in from their business services that use their business data. When used correctly big data can help increase the primary revenue streams for the company or even in some cases generate new ones when it is used to recognize patterns, predict customer behavior and provide their customers with added value services in addition to the core service provided. For example, these added value services can be statistics such as predictions or historical trend data. Sometimes, big data can even become a whole new revenue stream for the company depending on their business model, legislation, and the market they are in. When this happens, it may even change the company’s view of its customers/users as data can bring any number of new possibilities.

3.3 Utilizing Big Data

Big data is already changing our lives in countless ways in medical fields, online dating, sciences and in government to help run cities and countries. When considering big data utilization in business, the ways to utilize it depend mainly on the business model. In general, it is used to learn about their customers’ habits and preferences to cater to their needs while simultaneously using the same data for the company’s advantage (Marr, 2015).

Good well-known examples for these are companies like Facebook, Google and Twitter which give their users a seemingly free quality user experience, but in the background, they store, catalog, and analyze everything the user does so it can be productized in some way and then sold to third parties for profit. Currently, laws prevent most straight exchanges of customer data, but companies can circumvent these laws by creating products out of the users and their data for the third parties. Business models of specific companies affect how this can be achieved, but the most generic example of this is targeted advertising. Targeted advertising is something advertisers prefer because it provides them a way to reach their target audience without paying for the non-target demographic who are less likely to need their services and products (Facebook Ads for Business, 2018). An excellent example of targeted advertising is advertisers advertising to people in a certain geographical location, for example, advertising technology products in Las Vegas during the yearly Consumer Electronics Expo (CES). This allows advertisers to reach their intended target audience and making advertising more cost-effective. This type of advertising was a complete paradigm shift from the old way of advertising where advertising was done on television, radio, newspapers, roadside signs or any other static form of advertising medium. Utilizing these media resulted in the adverts to be more or less visible to everyone, regardless of gender or interests as it was not possible to limit the demographic.

To give a better understanding on what big data is and how its utilization in business situations functions, here are few well-known example cases. Large multi-national companies such as Google and Facebook provide free services to consumers by turning consumers into their products. They provide these free services to consumers by collecting all their data, analyzing it, selling it or productizing it as well targeted advertising to their advertisers'. This targeting is done with enormous success because of the users' tendency to share everything about their lives through these services. As the utilization of these kinds of techniques increases, the revenue of big data is predicted to reach $187 Billion by 2019 as new services created and businesses established, and old ones grow (IDC, 2016).

3.4 Example: Debt Collection

While Facebook uses big data for targeted advertising, other more specialized companies can also use it in their specific ways. Suitable examples of these more specialized companies are invoicing and debt collection companies that function in a highly-specialized and regulated field with highly customized business applications which in turn have access to highly valuable financial data. These companies can use different methods profile their debtors and give their customers unique insights about them. Debtor profiles can, for example, contain information about the debtor's paying and spending habits, the average term of payment, average purchase totals and general responsibility. In some cases, this can only be achieved in the relation between one customer and their debtors, but the real power of data analytics comes when debtor profiles are compared across different customers, and the patterns can be more easily recognized.

These added value services can make a substantial difference when customers are making decisions about their debt collection agency, but more importantly when they make decisions about giving credit to their customers.

4 CHALLENGES

There are challenges related to the Finnish electronic invoicing standards that create the need for recognizing names as the names are not adequately granular standardized. More fine-grained standardization could prevent most of these issues, and this could very well happen with the future standards.

4.1 Multiple Data Exchange Formats and Standards

There is an electronic invoicing standard format widely used in Finland called Finvoice. One of Finvoice’s most substantial issues is that there are four versions of it available on the standard’s website and some companies/software houses add their small custom changes on the standard or the systems do not produce utterly valid Finvoice formatted invoices. In addition to Finvoice, there are other formats created by different market leaders and software houses. Commonly Finvoice, as well as the other formats, have the data that is needed to print out and mail the invoice, but not much anything else. They were not designed with data in mind, but according to the business process.

4.2 Non-standardized User-supplied Input

As previously mentioned the in-coming data in the electronic invoice is not standardized to low enough level. For example, the field ‘name’ is left as a general name field in the Finvoice standard. This field can contain anything from ‘Niklas Lindgren’ to ‘VISMA PPG Oy / Niklas ja Essi Esimerkki’. There is no differentiating factor between a name of a business or a person, and sometimes it can even contain names of multiple persons or companies or even combinations of these. In general, user-supplied input can be one of the hardest data in data analytics to decipher. Without sanitation, the data cannot be used for more advanced purposes.

4.3 Customers Simply Do Not Care

Customers do not care about their data formats enough to keep them standardized. To be honest, data normalization is not even on the radar for most of the businesses for a good reason: It does not bring any additional income, and they do not need it if it does not affect their business processes. Sometimes a twist might even be necessary for them. For example, two names in one field could be an adequate solution for renters or other companies that sell to multiple people at the same time. Sometimes these issues are caused by an old system that might have been replaced at some point in the past, but no one bothered to correct the old data causing the new system to have data in multiple different formats. These are not real issues for the businesses because usually their software vendors might have done customizations for them to hide the issues after the fact or the businesses could have just adapted. Historically, invoicing services have just been mailing the letters out as-is without any further logic because there has not been any need for it, but as data becomes more critical for everyone involved additional logic is needed.

5 TOOLS

The programming language and the platform used for development and deployment are something that needs to be chosen carefully as it can lead to unforeseen issues and restrictions as the time goes on and the project grows. A clear set of criteria and a good comparison is used to define the optional platform and programming language for this project.

5.1 Programming Language

The choice of appropriate programming language is essential to any software project, and there are many issues to consider while one is making a choice. For this project, there were no specific requirements made by the commissioner. In fact, the commissioner encouraged to choose something that was not used much in-house, but of course still advised to keep it appropriate for the project.

5.1.1 Options and Criteria

Options for the programming language were considered with the help of IBM recommendations for selecting the optimal language. IBM's recommendations were used as a rough outline for selection of the language. IBM recommends to consider the following five factors:

1. The targeted platform
2. The elasticity of a language
3. The time to production
4. The performance
5. The support and community

(Source: IBM, 2011)

5.1.2 Comparing Languages

The language options were narrowed down to few potential ones based on the factors above and their use in-house software products. These language options were chosen to be Golang, Python3, PHP, and Adobe ColdFusion. Of these options, PHP and Adobe ColdFusion were discarded immediately because of their web development centricity as they cannot be easily daemonized. No official documentation exists for PHP how to achieve this (PHP Manual, 2018).

The targeted platform of the project was agreed upon to be either Linux or a Windows system, but Linux was chosen to be the preferred option because of its free availability and need for fewer hardware resources than Windows. Both of the languages considered have many similarities, but plenty of differences.

Python3 supports both of the targeted platforms (Python Operating Sytems List, 2018). Linux deployment is more convenient because of package managers commonly containing Python itself and usually the case-specific C and C++ header files that can be challenging to deal with on Windows systems. Python3 supports wheels which are in their simplest forms pre-packaged installers for Python packages that have all the binaries already compiled inside in such a way that the destination system does not need the specific header files installed as there is no compilation required. The main issue with these Wheels is that they are not available for every package as they require time and effort from their maintainers to create. There are significant security considerations to be made when running libraries and packages obtained from third-party sources on business systems as they can be infected with malware or other viruses. There are incidents in the recent past of infected packages being in widely used repositories. In 2016, NodeJS’ package repository npm had an incident of around 40 packages infected with malicious code (The npm Blog, 2016).

Golang also supports both of the target platforms (Golang System Requirements, 2018). As with Python3, Linux is a more natural system for deployment when specific C and C++ header files are needed because of Linux’s integrated package managers. Unlike Python3, Golang is a language that needs to be compiled into a binary executable before runtime. This makes precompiled extensions impossible. Thus, because Python3's wheels are not available for Golang, Linux became a better option for Golang.

Both of the languages are more conveniently deployed on Linux systems because of the availability of tools for deployment such as Docker, Jenkins, and Puppet that run natively on Linux. In theory, Windows is not a bad option, but in practice, it is not the ideal one because of lack of package managers and the fact that not all packages support Windows. Both of the languages favored Linux in the comparison making it the platform of choice.

The elasticity of the language for both of languages was chosen to be measured by the adaptability of the programming language to different workloads and by extension the number of packages available for the language. For Golang the number of packages in the official repository is well under 200 at the time of writing when Python’s same number was at over 130 000. It was assumed that Python would most likely be the more prefered language for developement because of its larger number of packages. This assumption was further confirmed by the fact the technologies preliminary thought to be used with the platform had better documented and maintained packages on Python3 than on Golang.

Performance-wise it was clear from the beginning that Golang would be considerably faster than Python3 because of it being closer to hardware. Benchmarks show Golang can be over hundreds of times faster while doing certain tasks (Go vs. Python3, 2018). Python3 could not compete with Golang on performance.

Support and community turned out to be a hard metric to measure as Golang has no official community forums whereas Python does. It was decided to measure the support and community activity by the percentage of Stack Overflow questions month-by-month as it is a popular platform for software developers who are looking for assistance with its 9.3m visits each day (StackExchange). If measured by the number of Stack Overflow questions per month in Diagram 1, Python was a clear winner (Stack Overflow, 2018).

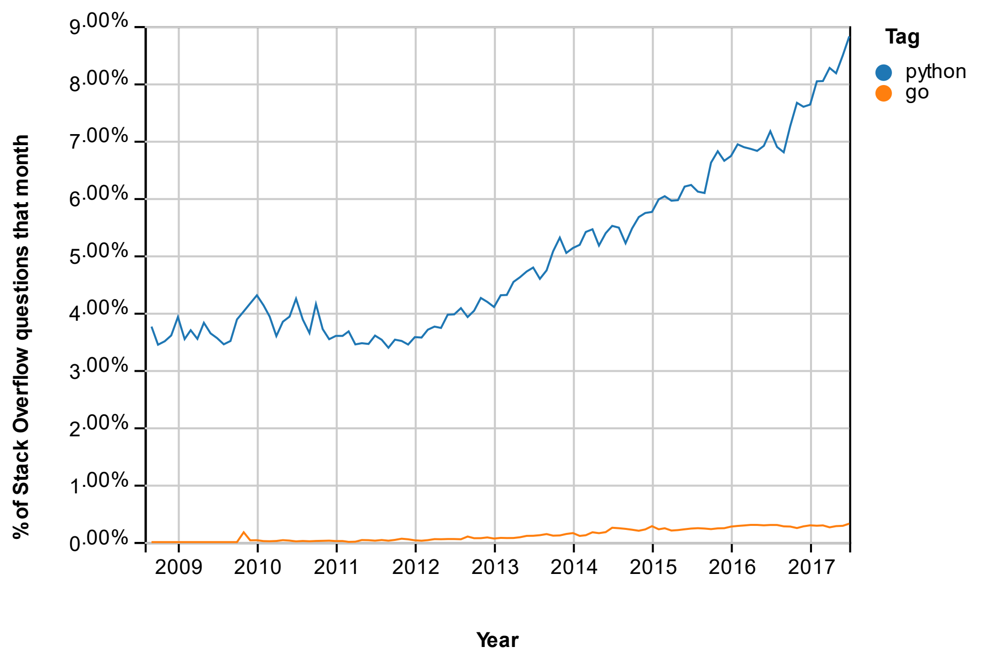

Diagram 1. Stack Overflow trends.

After looking at the programming languages from different perspectives, Python3 turned out to be the best candidate for the platform itself (see Table 1). Golang's performance turned out to be higher than assumed initially making its case clear, but in most cases, the performance benefits would be unnecessary, or they could slow development down too much. It was decided to have Python3 as the primary language of the platform as well as for the name parser component. It was also decided to keep options open to use Golang or any other language where it made more sense either performance or feature wise than Python3.

Table 1. Summary of the comparison of the programming languages.

Criteria / Language	Python	Golang
Targeted platform	X	X
Elasticity of the language	X	
Time to production	X	
Performance		X
Support and community	X	

5.2 Third Party Data Sources

Third party data sources are important when it comes to recognizing names. It is not feasible for companies to have their own dictionaries nor is it possible for companies to have information on every business in Finland without using third party resources. It is important to choose good quality data sources for the parser to reach the best possible results.

5.2.1 Institute for the Languages of Finland (Kotus)

Kotus as the official institute for managing the Finnish language published material t. As an institute for language research, they publish multiple resources related to the Finnish language including dictionaries. In the case of recognizing names out of a possibly arbitrary string, these dictionaries can be useful resources in either reducing the possibilities or helping in the detection itself. These dictionaries are at the very best a good, but not excellent, solution to the problem are not complete and are updated infrequently. For example, the Finnish dictionary was last updated in 2017.

5.2.2 The Business Information System (BIS)

The business information system is a joint service managed by the Finnish Patent and Registration Office and the Finnish Tax Administration (YTJ, 2018). Its purpose is to serve as one place to register new companies or access their information instead of going to the two separate government institutions. The business information system provides a free to use Application Programming Interface (API) for querying company details by a business id. This access is used to figure out the name of the business out of the name string in some cases where a business identifier is provided.

5.2.3 Population Register Centre

Population Register Center is a government organization responsible for maintaining the Finnish Population Information System, and they provide this data as a service to government and different companies (Population Register Centre, 2018). They also publish many statistics related to people and especially people’s names. These yearly statistics provide every first name and name-holders by gender as well as the same for last names. Only names that have more than ten name-holders in Finland are listed, but it does cover most Finnish and common foreign names.

6 NAME RECOGNITION LOGIC

In this chapter, every phase of the name recognition logic is explained in depth with visual representations of the process. Examples are also provided for easier understanding.

6.1 Overview

To better understand the logic of the parse, it has been split into four phases (see Diagram 2). These phases have their purposes but heavily rely on the data received from the previous phases. These phases are:

- Tokenization: Processes, normalizes and tokenizes the input.
- Classification: Classifies the tokenized input for better and more accurate results.
- Names of businesses: Recognition of names of businesses. This is an optional phase.
- Names of persons: Recognition of names of persons.

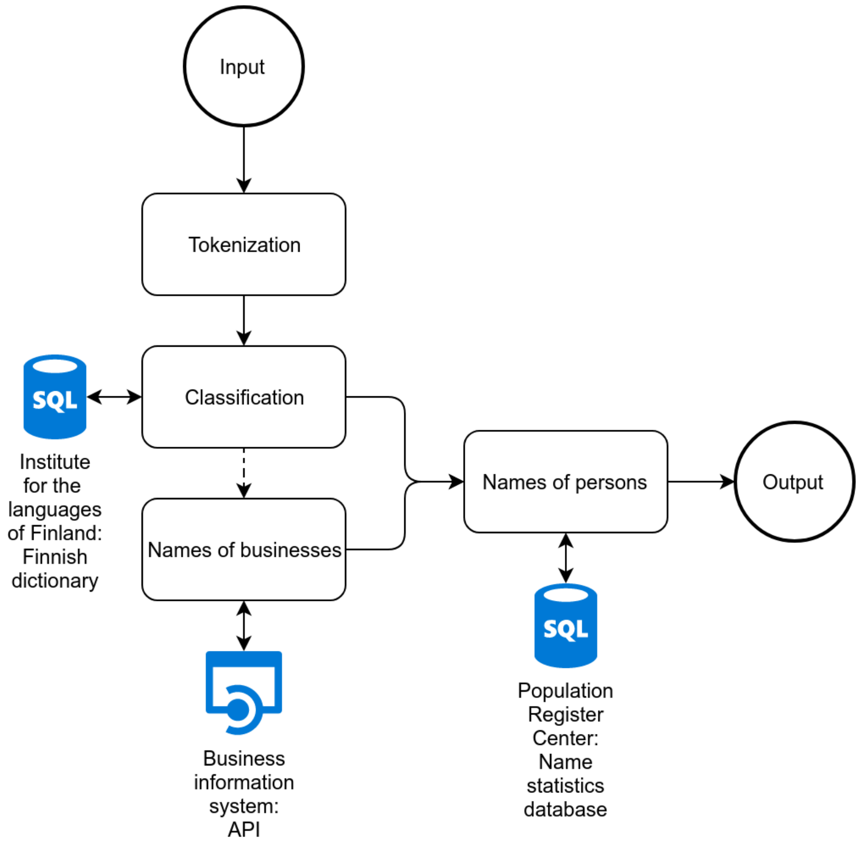
Input
Tokenization
SQL
Classification
Institute for the languages of Finland: Finnish dictionary
Names of businesses
Business information system: API
Names of persons
SQL
Population Register Center: Name statistics database
Output

Diagram 2. Overview of the logic.

6.2 Phase I: Tokenization

The first section of pipeline is where the input string is first processed. The purpose of this phase is to sanitize, normalize and tokenize the input thus transforming it into a standardized format more convenient for the rest of the pipeline. To sanitize and normalize the input, the string it is converted into its lowercase representation. Lowercasing is performed as capitalization is considered irrelevant for the recognition of names as correct capitalization is not guaranteed in the input. After lowercasing, next step for the string is to run a through Python3's unidecode function found in the unidecode module. Unidecode function strips the string of any non-ASCII characters and does its best to replace them with their closest ASCII representation. Non-ASCII characters are removed because they introduce additional variables to the logic but are not necessarily relevant for the sake of the output.

After these processes, the string is tokenized by splitting it by any whitespaces and special characters such as '/', '&', ','. This process results in an output list with every normalized part of the string as its member (see Table 2).

Table 2. Example input and outputs of the first phase.

Id	Input string	Resulting output
1	Niklas Lindgren	['niklas', 'lindgren']
2	Lindgren, Niklas	['lindgren', ',', 'Niklas']
3	VISMA PPG Oy/ Niklas Lindgren	['visma', 'ppg', 'oy', '/', 'niklas', 'lindgren']
4	Auto Aki Oy	['auto', 'aki', 'oy']
5	Niklas ja Essi Esimerkki	['niklas', 'ja', 'essi', 'esimerkki']

6.3 Phase II: Classification

The second phase of the pipeline is the classification phase. The purpose of this phase is to give each of the tokens from the previous section a classification if at all possible. This is carried out by checking the components against word lists to prepare them for classifying the names out of the components.

6.3.1 Delimiters

Delimiters inside the tokenized input need to be specially categorized to support needed flexibility further down the pipeline. Delimiters are a set of characters and words in the tokenized input. In the scope of this thesis, the used delimiters are: ‘ja’, ‘/’, ‘,’, ‘.’, and ‘&’. These delimiters are further divided into two subsets that are hard delimiters and soft delimiters.

Hard delimiters are defined as delimiters that are usually used to separate two completely independent parts of a string from each other. These hard delimiters, in turn, split the tokenized input further down into independent sections. Defining the hard delimiters is something that may heavily depend on the use of the input string, but in this specific case hard delimiters are ‘.’ and ‘/’. A good example of the use of hard delimiters is the input string ‘VISMA PPG Oy / Niklas Lindgren’ where the delimiter ‘/’ divides the input to two sections where one contains the name of the business and the other one the name of the contact person.

Soft delimiters are defined as delimiters that usually separate two parts of the string from each other, but both parts may have something to do with each other. These parts are divided into independent sections as the with hard delimiters but are considered to have a more flexible separation between the sections next to it as they do not impose any hard limits further down the pipeline. Soft delimiters are as follows: ‘ja’, ‘,’ and ‘&’. A good example of soft delimiters is the input string ‘Matti ja Maija Meikäläinen’ where the soft delimiter ‘ja’ divides the string into two fuzzy sections where both Matti’s and Maija’s last names should be recognized as Meikäläinen. Another good example is ‘Lindgren, Niklas’ where the soft delimiter ‘,’ is used as a separator between the first and last names but it should be considered as one whole name instead of two partial ones.

6.3.2 Blacklisting

The word blacklisting is used to prevent known common words being considered as names. As even a blacklisted component can be needed further down the pipeline, it is important to note the reason why the component was blacklisted and accurately classify it. There are multiple different categories of words and alike that are blacklisted. First of these classifications are the business types. These business types include entries such as the ones listed in Table 3. These are marked as business types and are mostly ignored further down the pipeline. The second category is the common custom words that are either widespread words such as names of cities, regions, or other common words in the input string that are known not to be valid names for people. This category should be carefully curated not to contain words that could also be names of persons as that could hurt the accuracy of the output. The third category is the Finnish words that are not nouns in the Finnish words dictionary according to the Institute for languages of Finland (see Table 3). By utilizing this dictionary, the accuracy can be further increased and the options for the names of persons reduced.

Table 3. Examples of custom blacklist entries.

Word	Classification	Word	Classification
oy	Business type	turun	Common word (custom)
tmi	Business type	helsingin	Common word (custom)
ltd	Business type	varsinais-suomen	Common word (custom)
as	Business type	auto	Common word (Kotus)
ky	Business type	kissa	Common word (Kotus)
ab	Business type	koti	Common word (Kotus)
osakehyhtiö	Business type	mopo	Common word (Kotus)
kommandiittiyhtiö	Business type	sieni	Common word (Kotus)
avoinyhtiö	Business type	autokauppa.	Common word (Kotus)

6.4 Phase III: Names of Businesses

The third phase is where the names of businesses are recognized. Third party APIs from the Business Information System are used to obtain the information of the business identifier in such a way that the names can be recognized out of the tokens. As a business identifier is required for the API queries, this phase is bypassed if no business identifier was supplied.

6.4.1 Business Information System APIs.

Business Information System APIs are a good source for requesting information on a specific business by their business identifier. The main throwback of using a third-party API is that the queries can take anywhere from 500 milliseconds to 1 second to resolve, thus making caching important for performance reasons. Because of the type of the material parsed, the importance of caching is emphasized even more because the parsed material can contain the same name multiple times in a row as invoices are usually sent in bulk. These API results also return different information in addition to official names, including names of board members, addresses, phone numbers, email addresses and tax debt. Some of these can be used elsewhere in the system for similar purposes, but here the focus is on the official names of the business.

6.4.2 Matching

When matching the input string to the official name(s) gathered through the APIs, it is essential to have all the data normalized the same way. Thus the API name result(s) are processed through the split and categorize sections of the parser before being matched.

After normalization, a score ranging from 0-1 is calculated between the first part of the name received through the API and parts of the input string excluding any delimiters and business types. In Diagram 3, this process is done to two examples. In the first one, the input string is correctly typed and in the second one, there is a typing error.

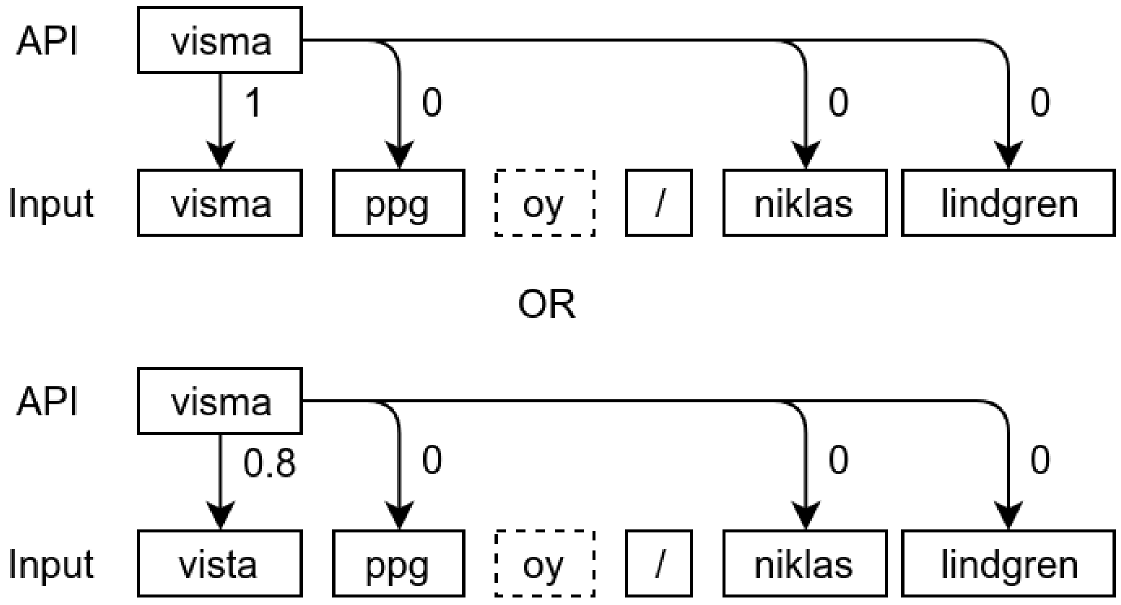

Diagram 3. Illustration of the process of detecting the first part of the name.

The highest score received from the previous phase is used as the beginning of the name, and the scores are calculated for the subsequent words until the name from the API request is exhausted. Business types have been specified as special cases because they are often left out of the names of the businesses when used, but if one does exist, and is of a correct type, it is counted as a favorable modifier.

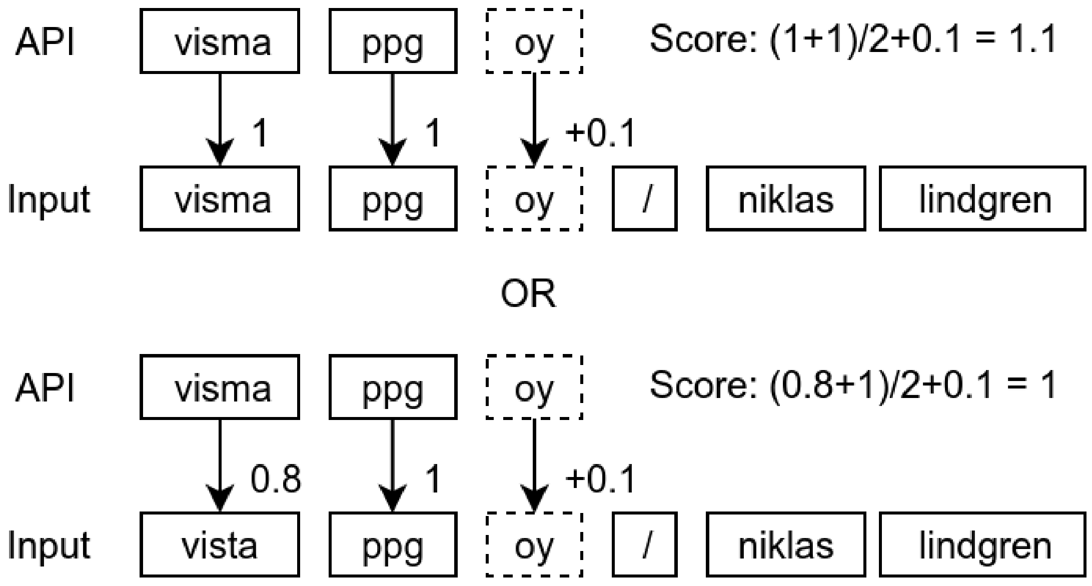

Diagram 4. Scoring of a business name.

After exhausting the name from the API, an average of the scores is counted by calculating the average score of all the parts of the name received from the API excluding delimiter and business type parts (see Diagram 4). Delimiter parts are entirely ignored to simplify the process, but in case of a matching business type part, the name is given a 0.1 score bonus bringing the maximum score a business name can receive to 1.1. This process is performed for all the official names received through the API query and the highest match above the arbitrarily chosen limit of 0.89 is chosen as a match. The matching part of the input string is then categorized as the name of a business, and the name recognition process continues.

6.5 Phase IV: Names of Persons

The fourth phase is for finding names of persons out of the tokenized and classified input from which additionally a business name has already been recognized. For recognizing names in the input, the People Register Centre’s statistical database is used.

6.5.1 People Register Centre’s statistical database

Both first names and last names are going to be detected based on the People Register Centre’s name statistics. These statistics do not have every name in use in Finland nor can the process recognize names correctly all the time, but the goal accuracy is best effort. To increase the chances of recognizing the names the statistical database has been standardized and normalized the same way as the input string.

6.5.2 Matching: A Simple Case

A first example case is the previously used input string of ‘VISMA PPG Oy/ Niklas Lindgren’. First, each of the non-parsed parts of the tokenized input is searched through the statistical database. In the example, in Diagram 5, both entries match both first and last names, but there is a clear separation between them as ‘Niklas’ is a more popular as a first name than as a last name, and ‘Lindgren’ is a more popular as the last name than as first name. ‘Niklas’ being more popular as the first name and ‘Lindgren’ being more popular as the last name they are processed as so and given out as the output along with the business name.

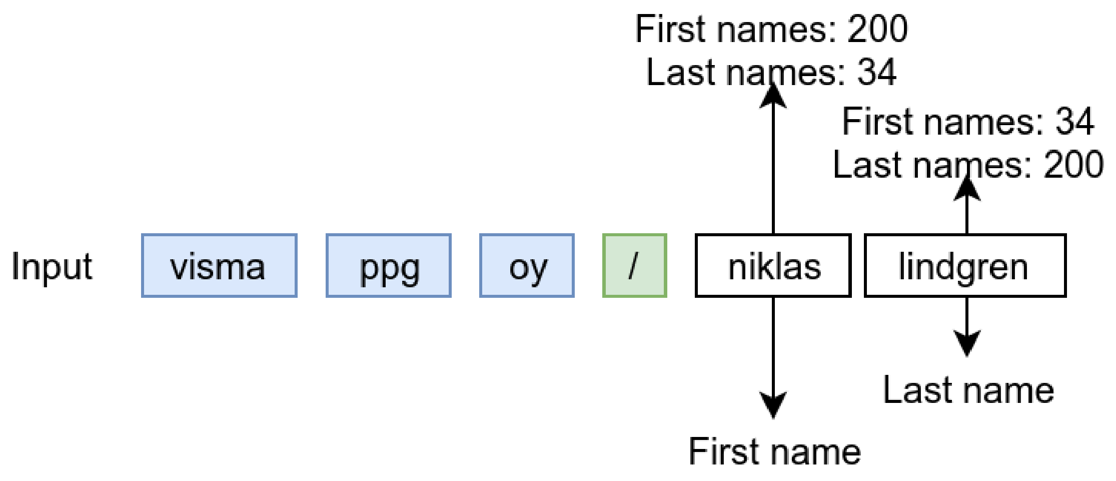

Diagram 5. Illustration of a simple case with arbitrary values.

This results in the following output:

- Businesses:
  - Business name: VISMA PPG OY; Business identifier: XXXXXX-Y
- Persons:
  - First name: Niklas; Last name: Lindgren

6.5.3 Matching: Soft Delimiter

For matching, cases with soft delimiters similar to the Diagram 6 first the unparsed entries of the list are checked through the statistical database. After that, the names are determined for both groups as normal. After determining the in-group names, both groups are checked to contain both first and last names. Group 0 contains only a first name, and group 1 contains both.

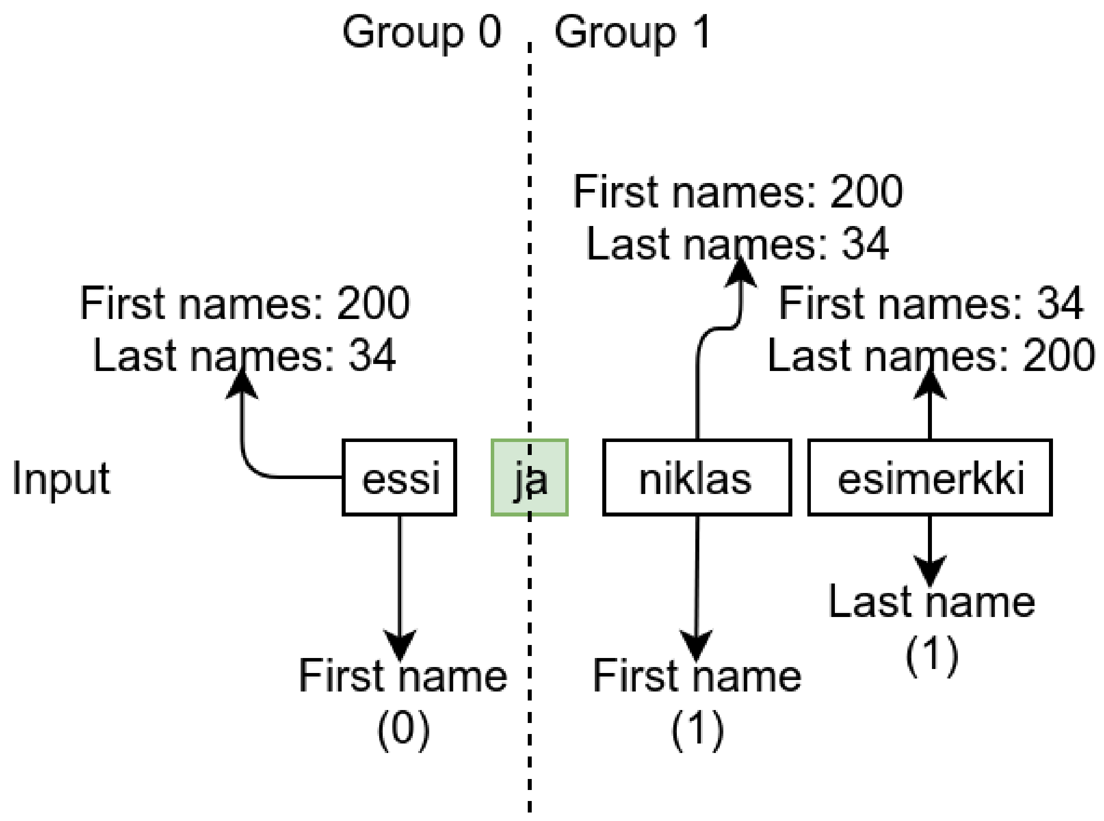

Diagram 6: Soft delimiter example.

In this case, for group 0, a search is initialized to find the closest complementing value which in this case is the last name from group 1. This search will only go through soft delimiters and will pick the closest complementing value (see Diagram 7).

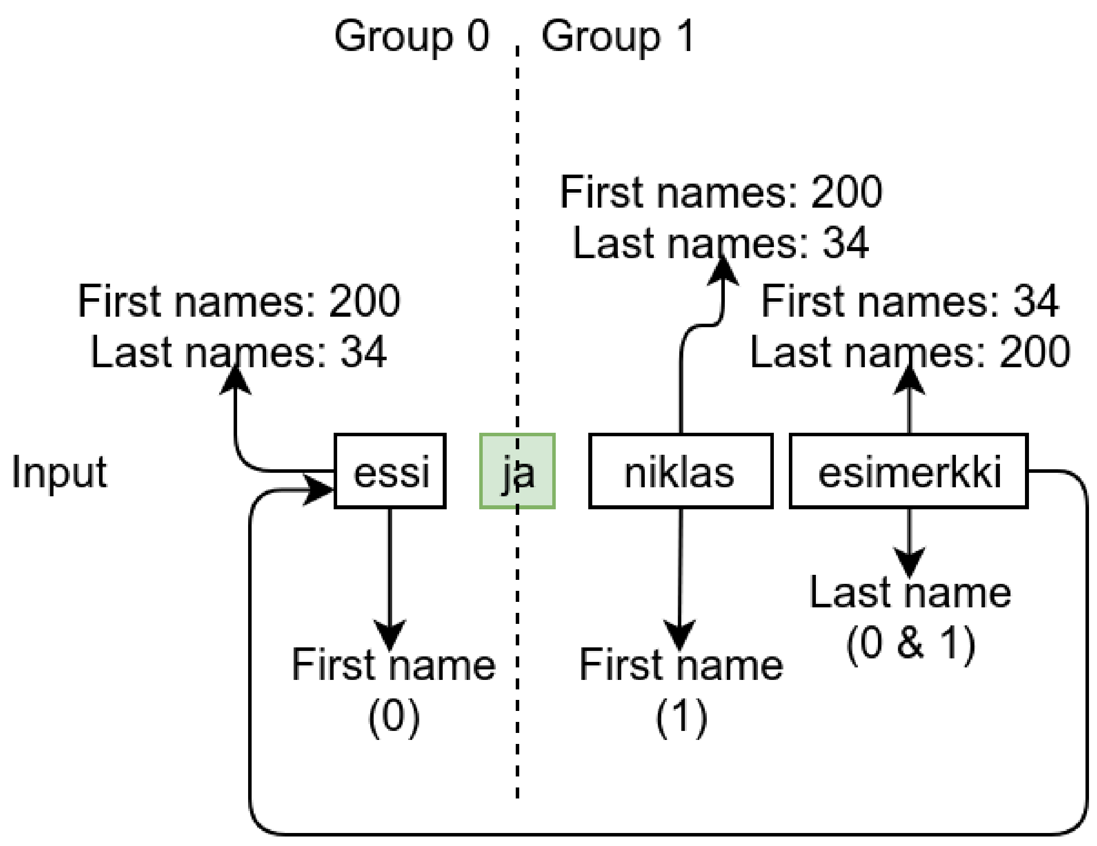

Diagram 7. Crossing the fuzzy soft delimiter.

This results in the following output:

- Persons:
    - First name: Niklas; Last name: Esimerkki
    - First name: Essi; Last name: Esimerkki

6.5.4 Matching: Hard Delimiter

For matching with a hard delimiter, the example input string is ‘Auto AKI / Niklas Lindgren’ and no provided business identifier. The word ‘auto’ was previously blacklisted with the Finnish dictionary and flagged not to be matched. As with the soft delimiters, both groups are first processed through the statistical database independently. The same process as with the soft delimiters is used to find a suitable last name for ‘Aki’ in group 0, but as there is no other group separated by a soft delimiter, it cannot complete (see Diagram

8).

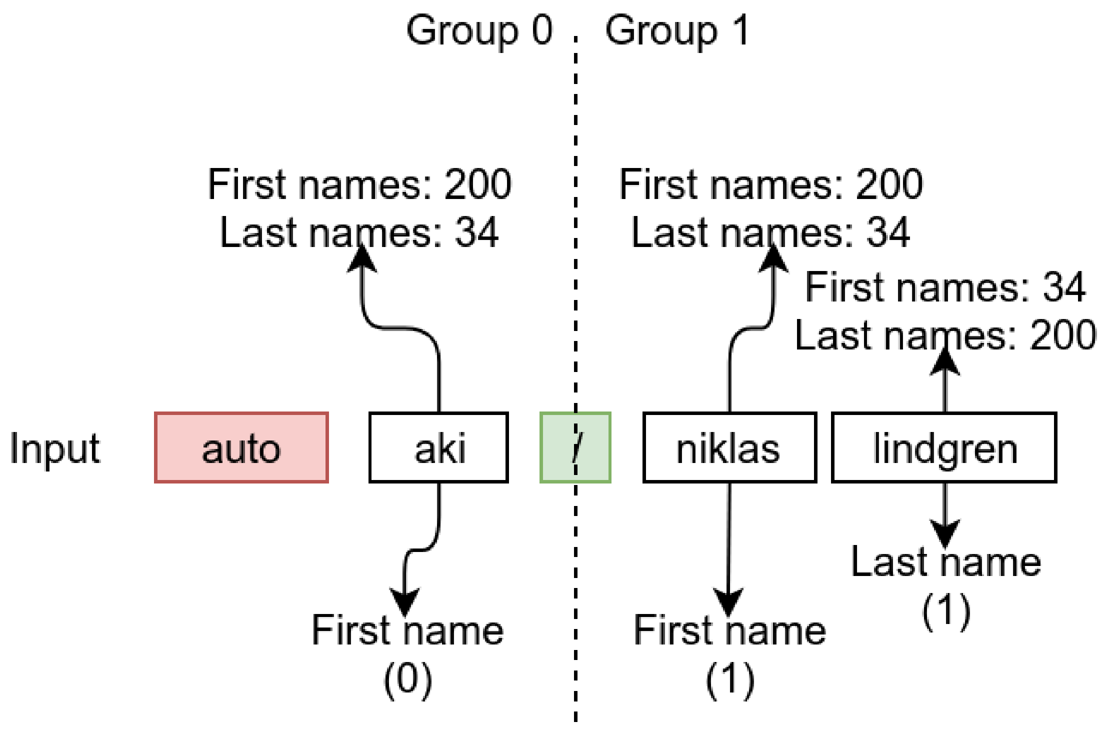

Diagram 8. Hard delimiter illustration.

This results in the following output:

- Persons:
  - First name: Aki
  - First name: Niklas; Last name: Lindgren

7 CONCLUSIONS

The commissioner of the thesis had a need to recognize names out of a string field for faster processing and automation. This thesis created the necessary software to solve this issue by comparing different programming languages and by developing the necessary software logic.

As the use of big data continues to grow in different fields not traditionally focused on it, there will be more and more instances where recognizing details from unorganized data streams or fields is necessary. It is important to be able to sanitize and standardize different kinds of data even if the quality of the result is only best effort. More important than quality is standardization because it allows connecting different data points even if it is not necessarily known what the data points exactly are, but as long as the standardization is precise enough other data points can be used to strengthen the result. In the case of recognizing names out of a string, this could be the case of not recognizing person’s last name but still recognizing their first name. Even if the data is not complete, this data can be used with other data points to increase the overall accuracy of the data.

In this specific use-case, for name recognition, the accuracy is relatively high with simple to complex name formats. Accuracy depends on the data and statistical databases used, but in tests the name recognition was able to reach 95% accuracy. It is important to have good and comprehensive data sources for the recognition and matching procedures to use. The most significant issue for the recognition are names that are either relatively new or unused in Finland resulting in them not being included in the name statistics. There are multiple solutions to this issue. One of them is flagging unrecognized strings for human intervention as humans can recognize names with just intuition. As this is not a scalable solution, finding name statistics for other world counties is a better more scalable solution. It will not stop the need for human intervention but should reduce it considerably. For example, Swedish name statistics are a great source for middle-eastern and German names but implementing these is not easy as Sweden does not allow easy export of the data out of their system.

For now, the accuracy of the name parser reaches the goals set for it, but there are already multiple new features and improvements under consideration. The previously mentioned foreign name statistics being one, but more importantly a reverse company name search without a need for the business identifier. Both are expected to increase the recognition accuracy dramatically.

REFERENCES

About Facebook ads, Facebook, visited on 30.3.2017 https://www.facebook.com/ads/about/?entry_product=ad_preferences

98 different data points that Facebook uses to target ads on you, visited on 30.3.2017, Washington Post, https://www.washingtonpost.com/news/the-intersect/wp/2016/08/19/98-personal-data-points-that-facebook-uses-to-target-ads-to-you/?utm_term=.5b81875e54e8

Worldwide Big Data and Business Analytics Revenues Forecast to Reach $187 Billion in 2019, According to IDC, 23.5.2016 https://www.idc.com/getdoc.jsp?containerId=prUS41306516

Marr, Bernard. Big Data, edited by Bernard Marr, John Wiley & Sons, Incorporated, 2015. ProQuest Ebook Central, https://ebookcentral.proquest.com/lib/turkuamk-ebooks/detail.action?docID=1895824.

General Data Protection Regulation, Council of the European Union, Visited on 11.4.2018, http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf

Facebook ads for business, Facebook, Visited on 10.4.2018, https://www.facebook.com/business/products/ads

PHP Manual, Mehdi Achour, Friedhelm Betz, Antony Dovgal, Nuno Lopes, Hannes Magnusson, Georg Richter, Damien Seguy, Jakub Vrana, and several others, Visited on 10.4.2018, https://secure.php.net/manual/en/index.php

Python Operating Sytems List, Python Software Foundation, Visited on 10.4.2018, https://www.python.org/downloads/operating-systems/,

Stack Overflow trends, Stack Overflow, visited on 8.4.2018, (https://insights.stackoverflow.com/trends?tags=python%2Cgo

Golang System Requirements, Unknown, Visited on 10.4.2018, https://golang.org/doc/install

‘crossenv’ malware on the npm registry, The npm Blog, Visited on 10.4.2018, http://blog.npmjs.org/post/163723642530/crossenv-malware-on-the-npm-registry

StackExchange statistics, StackExchange, Visited on 8.4.2018, https://stackexchange.com/sites?view=list#traffic

Go vs Python3, The Computer Language Benchmarks Game, visited on 8.4.2018, https://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=go&lang2=python3

Population Register Centre, Population Register Centre, visited on 12.6.2018, http://vrk.fi/en/about-us

What is BIS?, The Business Information System, visited on 12.6.2018, https://www.ytj.fi/en/index/whatisbis.html

Selecting the optimal programming language, Jerry Reghunadh and Neha Jain for IBM, 2011, visited on 7.2.2018, https://www.ibm.com/developerworks/library/wa-optimal/