Project Creation - Final Report

Pedram Ardeshirzadeh    Kevin Kessels    Ping Wan

July 9, 2014
ABSTRACT

CHAINels is a business platform where companies can keep each other updated about the latest news and collaborate more efficiently with each other. Companies often form groups centered around common aspects of their businesses. In order for these groups to be part of this platform, each company needs to have a profile. Data about each company that is to become part of the platform is provided to CHAINels by the group. CHAINels then manually creates the profiles for each of the companies. This means manual verification of all the data that needs to be entered in CHAINels’ system. With more companies using the CHAINels platform, this manual process becomes very time consuming and less profitable.

For this problem CHAINels requested a system that can automate this process. Our system uses external sources to classify and verify data fields from a CSV file, provides the user an interface with the possible company profiles and allows the user to manually verify the information before finalizing the creation of the profiles. Eventually the group representative or the companies will use this system themselves.
PREFACE

For the completion of the bachelor, students are required to do the TI3800 Bachelor project. This is the final report which includes the assignment, research and development done by a team of three students. The assignment was issued by CHAINels, which is a start-up based in Delft. We would like to thank CHAINels for providing us with an interesting assignment and for the excellent support during the orientation and development phase. Finally, we would like to thank Dr. Alessandro Bozzon for his guidance and advice, which helped us focus on what was most important.
CONTENTS

1 Introduction 6

2 Problem Analysis 7
  2.1 Problem Description ........................................... 7

3 Requirements 9
  3.1 Functional Requirements ........................................ 9
  3.2 Nonfunctional Requirements ................................. 10

4 Design 11
  4.1 Current System .................................................... 11
  4.2 Back-end ............................................................ 12
    4.2.1 Extraction and Classification ................................. 12
    4.2.2 Company Website Scraping .................................... 14
    4.2.3 Verifying the Data ............................................... 17
  4.3 Front-end .......................................................... 17

5 Implementation 19
  5.1 The Controller ..................................................... 19
  5.2 Creating The Companies .......................................... 19
  5.3 Extraction From CSV Files ........................................ 22
  5.4 Classification ....................................................... 22
    5.4.1 API Handlers .................................................... 23
  5.5 Scraper ............................................................... 23
    5.5.1 The Scrapy Scraper ............................................. 24
    5.5.2 The Scraper Handler ........................................... 25
  5.6 Verification ........................................................ 26

6 Quality Assurance 28
  6.1 Process Quality .................................................... 28
    6.1.1 Agile .............................................................. 28
    6.1.2 Version Control ................................................ 28
  6.2 Testing ............................................................... 28
    6.2.1 Unit Testing .................................................... 29
    6.2.2 Performance Testing .......................................... 29
  6.3 SIG ................................................................. 30
    6.3.1 Feedback ......................................................... 30
    6.3.2 Improvements .................................................. 32
7 Evaluation of the Requirements
   7.0.3 Functional Requirements ....................... 33
   7.0.4 Nonfunctional Requirements ................... 33

8 Recommendations ........................................... 35

9 Conclusion ..................................................... 36

Appendices ....................................................... 37

A Problem Description ....................................... 38

B Orientation Report ......................................... 39

C Plan of Approach ............................................. 53

D SIG Feedback .................................................. 61
   D.0.5 First Feedback ......................................... 61
   D.0.6 Second Feedback ...................................... 61
1 INTRODUCTION

CHAINels is an online social networking platform that was founded by computer science students Erwin Buckers and Vincent Koeman. Unlike other social networking platforms, such as Twitter and Facebook, where the majority of the communication is between individuals, CHAINels is strictly aimed towards businesses. Each business is a member of a business group. A business group can be a group of companies that are located in the same mall or any other union. These companies can keep each other updated about the latest news and events. CHAINels aims to provide a platform where businesses can collaborate better and more efficiently in the area of knowledge, safety, collective purchasing and policy.

In order to create a new business group, the companies have to be manually entered in the system. This manual process is currently done by the employees of CHAINels. Company data is provided, but the variety and completeness of this data makes it difficult to use. Therefore, missing information has to be collected and all the information has to be verified. All properties, such as address and website, need to be correctly entered in the system. In the beginning this was still both manageable and profitable, but during last few years the client base has grown rapidly and it is expected that the growth will continue. Therefore the manual creation of the business groups has been consuming more time and getting less profitable. For this problem CHAINels needed a more automated way of creating groups when provided a data file by the client.

For our project we developed an automated way of finding and verifying companies. It involves extracting the file for information as well as finding information from different sources like OpenKVK, GoogleMaps and company websites. Then all the information is cross-referenced to determine if it is correct. Finally, the most likely to be correct information is displayed for the user to perform a final check.
2 PROBLEM ANALYSIS

The CHAINels system is based on business groups. The members of each group are companies that are connected with each other in some way. They could all be located in the same mall or be members of the same union. Usually, the mall or union has a person that keeps track of all the companies and their information in some way. In our context we call this person the group administrator.

When the board of such mall or union has decided to use CHAINels' platform, the group administrator provides a file containing information about all the different companies from the group. CHAINels needs to process all these different administration files manually for two reasons: correctness and completeness.

Not all group administrators keep detailed records of the company. In some situations it is possible they only have the basic information such as name and address and do not have other detailed information about the companies. Detailed information is needed in order to take full advantage of CHAINels’ platform.

Another problem is that group administrators do not always keep their records up to date. Some properties of a company can be false or outdated due to a move to another building or the whole company no longer exists.

Currently the files are manually processed by employees of CHAINels. During the process they need to verify each property for all the companies. This means using location services such as Google Maps or Bing Maps to verify the address fields. By using various search engines they can verify or complement the other data fields as well. After the verification the companies and group are created. CHAINels performs this manual process in order to make the transition to CHAINels’ platform as easy as possible for the client. The biggest drawback of this process is time. Whenever CHAINels has acquired a new client, they need to spend a lot of time to manually verify and complement each company. Worst case scenario, this could take hours of work if the group contains a lot of companies, and information is incorrect or incomplete.

2.1 PROBLEM DESCRIPTION

Our task was to develop a system that automates the verification as well as the process of finding missing information. The user needs to be able to upload a file that the system will process. The system must show the com-
panies that it has found to the user for additional manual verification. The system will be used by both CHAINels’ employees and group administrators. Therefore, it also needs a graphical user interface for intuitive interaction. The interaction with the current CHAINels’ system will be minimal, because the profile creation system will be a separate entity.
3 REQUIREMENTS

This section discusses the requirements that need to be taken into account during the design of the system. The requirements are the result of communication with CHAINels and our coach. The requirements are divided in functional requirements and nonfunctional requirements.

3.1 FUNCTIONAL REQUIREMENTS

- Must categorize the data fields of a provided data file.
- Must categorize the data.
- Must be able to handle CSV files.
- Must obtain data from page with APIs.
- Must obtain data from pages with no APIs such as company websites.
- Must extract the company properties based on the company model.
- Should find data with the use of external sources for missing fields.
- Must match data found on external sources with given data.
- Must verify the correctness of provided data with found data.
- Should determine the accuracy and correctness of found data.
- Must mark data as unsure if conflicting data is found.
- Must mark data as unsure if no additional data is found.
- Should use SSL (Semi-supervised Learning) techniques to learn from new data.
- Should save the modified file for use as training data.
- Must show the user the modified data file for manual verification and editing.
- Must provide suggestions during user interaction based on verification process.
- Could scrape for company logo otherwise should provide a temporary logo.
3.2 NONFUNCTIONAL REQUIREMENTS

- Should be extendable to handle different kinds of postal code formats.
- Should be extendable to handle different kinds of address formats.
- Should be extendable to handle different kinds of phone number formats.
- Should be extendable to display different languages using CHAINels’ already existing language system.
- The system must have a high level of stability.
- The system must have a high level of testability.
- Could be extendable to handle other data file types.
- Time between user uploading data and displaying the verified and found data must be less then five minutes.
- Scraping information per company must be done within five seconds.
- Must detect missing information 100% of the time.
- Must work on popular browsers: Firefox, Chrome, Safari and Internet Explorer 7+.
- The user verification process should be intuitive.
- The system should provide feedback when user interaction is needed.
- Should be possible to add new extraction and/or verification techniques in the future.
- Should follow the Model-View-Controller architectural pattern.
4 DESIGN

In this section an overview will be given of the design of the system. First a short explanation of the current system is given. Then we will cover the back-end as well as the front-end.

4.1 CURRENT SYSTEM

The CHAINels system is built using the software architectural pattern Model-View-Controller (MVC). The model represents the knowledge of the system. It defines how objects are structured as well as the relationship between different objects. The most important and only model that we have to interact with is the Company model.

The company model (Figure 1) has the following important properties:

- A company name
- An about property, which holds information such as a small description of the company, business number and business hours.
- An address object, which holds information such as street name, street number and zipcode.

Figure 1: The company class
• An industry key and industry label, which we will ignore because clients will manually select available choices at a later stage during the profile creation.

• A contact object, which holds the email, phone number and website address.

• A company logo.

There are other company properties that CHAINels keeps track of, but these are not relevant for our program as they have to be handled by CHAINels’ own system.

The view consist of all user interface elements that are displayed to the users. CHAINels uses classes for each interface element. This supports the programming paradigm Object-oriented programming (OOP), which CHAINels incorporates through the whole system.

The controller handles the connection between the user interface and the back-end.

4.2 BACK-END

The back-end consists of multiple components working together to enable the processing of company information. Figure 2 shows the general flow of the system. First the data file is read and prepared for classification and verification. The data fields are then classified utilizing external resources. After the classification, temporary companies are made for each row. Each company will be verified and missing information will be filled in as much as possible. This process makes use of external resources and a scraper. The companies are then displayed to the user for additional verification and final confirmation.

4.2.1 EXTRACTION AND CLASSIFICATION

In order to be able to use the information provided by the client for creating the company profiles, the given data file needs to go through an extraction process followed by a classification process. This is important, because knowing what information has been provided will help with the verification later on. The extraction simply consists of translating the data fields of a file to a readable format for our programming language. Next comes the classification of the extracted data. In general, clients provide information
Figure 2: The flow of the system
such as the company names, addresses, phone numbers, e-mail addresses and websites. In order to classify all these different kinds of information, different techniques need to be utilized. This means using a combination of regular expressions and information acquired from external resources such as Google Maps, OpenKVK or OpenStreetMap.

For extendability reasons, it is best to not use regular expressions as the sole method for classification. These do not perform as well when there is no general international format, as is the case for zip codes, street names and street numbers. While external resources can also have international limitations, OpenKVK for example is only useful for Dutch companies, their main limitation lies within their availability. Most external resources either have a limit for the number of requests per second or the total number of requests per day. It is therefore best to have a combination of regular expressions and external resources for classification. Furthermore, it is important to note that the back-end is setup in such a way that it can be extended with other external resources in the future for international usage.

### 4.2.2 COMPANY WEBSITE SCRAPING

One of the ways that the system obtains information for the company profiles is through the use of a scraper. A scraper is a program that navigates through the web in a fashion similar to a user. This allows it to reach any place that a user can and does not require the website to have an infrastructure, such as an API, in order to obtain information from it. Afterwards the scraped information can be stored in a structured format for analysis. This form of scraping is known as web scraping. The scraper is used on a company’s website as these generally contain a large portion of the information required to setup a company profile. The flow chart in figure 3 and explanation below will show how the scraper works.

The scraper is called via the ScraperHandler class with a collection of arguments that will be used as parameters for the scraping process. The mandatory argument is the website to scrape. Optional arguments include company name, street name, zip code, etc. These optional arguments, if available, are then converted to regular expressions to be added to the build-in ones used to scrape the data. The scraper makes use of a link extractor that keeps track of which links still have to be followed and whether they lead to pages not yet visited. For each page the scraper uses XPath expressions to select elements from the page. These allow it to select all the text elements of the page and omit the formatting text, which contains no information the scraper is interested in. The scraper then removes extra
Figure 3: Flow chart of the scraping process
spaces as they can negatively effect the effectiveness of the regular expressions. The regular expressions are then applied and the results are stored in the Item Loader. The Item Loader class acts as a middleman between the data scraping and the data storage. Since the regular expressions can match more than one instance of, for example a street name, per page, the Item Loader allows for the merging of all found instances before creating an item for them. The same process is done for image scraping except that the XPath expressions are now used to select image elements of a page. The image name is extracted and tested to see if it is a potential company logo and if the image has not been scraped previously. After all the information of a page has been scraped, the Item Loader creates an item with it. When there are no more links to be followed a feed exporter outputs all the items into a CSV file.

Extendability and maintainability are important aspects of the scraper. The scraper was created using the Scrapy framework, which uses a modular structure allowing the previously mentioned software engineering aspects to be met. The scraper was designed to be able to handle as many shapes and sizes of websites. This was done by making many of the searching methods as generically applicable as possible. Besides text, the scraper also has a pipeline for processing images discovered on a website.

The scraper makes use of regular expressions to scrape information. The regular expressions used for scraping are very loose since the scraper’s goal is to find as many instances of each property as possible. This is possible because the information found by the scraper is not accepted until cross-referenced with the other external sources. In order to allow the regular expressions to work optimally the text of the website is normalized before it is checked using the regular expressions. This normalization prevents any unusual formatting of text from negatively affecting the regular expressions ability to recognize the information. Not all information to be scraped is equally difficult to detect. Certain information such as phone numbers and zip codes are far easier to detect in a block of text when compared to street names and company names. This is because company names are generally a series of words that is only distinguishable from the rest of the words in a sentence by semantics (meaning), and regular expression detect from a syntax (structural) point of view. The scraper is also able to search for company attributes that have already been provided by the client. This is what the optional arguments are used for. It performs this task simultaneously with the build-in regular expression matching.
4.2.3 VERIFYING THE DATA

The purpose of the verifier component is to use the information provided by the client in conjunction with the information gathered by the scraper to determine if the client has entered their information correctly. The verifier component is designed to use the large amount of information at its disposal as effectively as possible. The information is given weights depending on its source and how often the data was found. For example, in the case of company name, it would be given a greater weight if the specific name was found fifty times as opposed to it only being found ten times. The final total weights given to each version of a company attribute are then used to give an indication of the correctness of the information.

4.3 FRONT-END

The main focus of the project was the extraction of company data, verifying and also complementing it where possible. However, because of the usage of the system by not only CHAINels, but also the group administrators, the creation of an intuitive user interface was just as important. The following were important points to take into consideration while designing the user interface:

- How many companies do we display at the same time?
- How does a user move to the next or previous company?
- How do we display the confidence level of each property?
- How can a user edit the properties of a company?

At first a very simple mock was made of the user interface (Figure 4). In this mock there are two arrows which users can use to navigate to the next or previous company. An indicator bar is placed under each property in order to show how confident the system is about the value of that property. Furthermore, a counter is placed under each company to indicate how many companies the user still needs to review.

While developing the mock, we made the observation that clicking with the arrows can be a bit cumbersome when there is a large amount of companies and therefore created a second mock (Figure 5).

In this second version of the mock we removed the arrows and added a container that can scroll horizontally in order to move through the companies.
more quickly. The container can show three companies at the same time resulting in less scrolling. Also, instead of the indicator bars we used colored lines to indicated the confidence of a certain value:

- Green: probability > 75%
- Orange: 33% < probability < 75%
- Red: probability < 33%

After showing the second mock to the designer at CHAINels, he made the following remarks:

- When displaying more properties, the whole view can look very cramped.
- The layout does not follow the rest of the design of the website

Taking these remarks into consideration, it was decided to use the design of the editing page that CHAINels uses for existing companies. The functionality is very similar, but more importantly it increases the cohesion with the rest of the CHAINels’ website design.

![Figure 4: The first mock of the user interface](image_url)
5 IMPLEMENTATION

The implementation details and integration of each of the components as seen in Figure 7 will be discussed in this section.

5.1 THE CONTROLLER

The whole front-end is created by the CreateController class. Based on the request it receives from the user, it generates the desired HTML view to display. The create.js javascript file handles the interaction with the user and is the file that takes care of sending the correct requests to the CreateController. The HTML views that the CreateController correspondingly generates and returns as a response are also handled by create.js.

When a user uploads a CSV file, CreateController makes use of a ProfileCreator object in order to start the classification and verification processes. After these processes are finished, the user gets to manually verify and edit the results to finally create the company profiles.

5.2 CREATING THE COMPANIES

Before creating all of the company profiles, as mentioned in Section 4.2, the three processes of extraction, classification and verification need to be realized.
Within the ProfileCreator class, different methods realize these three processes. It is given a CSVHandler and an EntityRecognition object as its constructor parameters which are used in the `categorizeFile` method to perform the extraction and classification processes. With `createCompanies` a list of Interim Companies will be created based on the classified and extracted data. Interim Companies are extended versions of Companies which can be created with CHAINels’ own Company class. The InterimCompany class holds an array which has all of the confidence values of the company’s properties stored in it. The last method, `verifyCompanies`, handles the ver-

![Figure 6: Final design of the user interface](image-url)

20
Figure 7: The class diagram of the system
ification process and assigns the confidence values to all of the previously created Interim Companies.

5.3 EXTRACTION FROM CSV FILES

One of the simpler components, the sole purpose of CSVHandler is to read and extract data provided in a CSV file and return a two-dimensional array with the extracted data. In order to do so, it starts first with reading the whole file, separating each line. The number of lines will be the size of the first dimension of the array. The second dimension is then filled with the data of each line, extracted by looking for the separation delimiter used in the CSV file (usually ",";>). While doing so, it will keep in consideration that data fields can contain CSV special characters and therefore replaces any of those with safe characters to revert back at the end of the process.

5.4 CLASSIFICATION

After having extracted the data from a CSV file and having a two-dimensional array with the data, it needs to go through a classification process as discussed in 4.2.1. The classes that deal with handling the external sources, the API Handlers, will be discussed in Section 5.4.1.

The class that is the core of the classification process is the Classifier class. It is the link between the EntityRecognition class and the different API Handlers. It receives data from EntityRecognition that needs to be classified. The received data is subsequently provided to the different API Handlers that are to be used. Based on their results and usage of regular expressions, it answers to EntityRecognition whether the received data is a certain property or not. Besides providing the Classifier with data, the EntityRecognition class also keeps track of the classified data. Once a certain property has been classified, the array index of that property is mapped to the property name. In order to save time, the classified array index will be skipped in later classification iterations. This could backfire if a wrong property has been classified and therefore, to fight this, a certain classification order is used based on how easily a property can be classified. Further optimization is applied by temporarily storing the results of a certain query for future usage in order to not make an unnecessary amount of request to external sources with duplicate requests.
5.4.1 API HANDLERS

In order to easily make use of many external sources and for future extensions, the APIHandlerInterface interface is used for all of the API Handlers. Every API Handler class needs to have a method called `doQuery` that performs a query search and a method per property that needs to be classified. As an example, [INSERT FIGURE] shows the OpenStreetMapHandler class. The `doQuery` method shows how based on a query, data is requested from OpenStreetMap. The `onStreetName` method for example returns whether a given string is a street name or not.

5.5 SCRAPER

The scraper part of the system is composed of the Scraper itself and the classes which facilitate the interaction with the scraper.
5.5.1 THE SCRAPY SCRAPER

The scraper is implemented in Python using the Scrapy framework (http://doc.scrapy.org/en/latest/). It is the only part of the program not written in PHP. The scraper is composed of two main files: the Item file and the Spider file. There are also several additional files for scraper settings, test cases, and start-up and external argument handling.

The Item file contains the CompanyItem class and the CompanyLoader class. The CompanyItem class defines a CompanyItem object with attributes corresponding to the company attributes the scraper will scrape for. The CompanyLoader class is responsible for storing the information obtained by the Spider in a CompanyItem object. The loader also performs formatting on the scraped information such as remove leading and trailing whitespace.

The Spider file houses the CompanyWebsiteSpider class which performs the scraping of information. The Spider class has regular expressions to search for the street name, zip code, city, phone number, and email of a
company. First the Spider uses XPath expressions to select all text elements of the page. Then the Spider formats this text by removing extra spaces. Lastly, it applies the regular expressions. Any matches are temporarily stored using the CompanyLoader class. Once an entire page has been scraped the information is formatted and stored in a CompanyItem. This means that an item contains all the matches of one page which allows the use of the item limit setting, discussed in the next paragraph, to limit the number of pages to scrape. The Spider also searches through the HTML of the website for image elements. From these the image name is obtained and used to determine if the image is a potential company logo. Extra care has been taken to prevent the downloading of the same image more than once. Any accepted images have there URLs stored in the CompanyItem of the page after which the ImagePipeline, which is part of the Scrapy framework, schedules and downloads the images.

The settings file contains the customizable options that the Scrapy framework provides. In this file the settings for exporting and storing the scraped information to a CSV file are set. It also contains the options for the Image Pipeline which handles the scheduling and downloading of potential company logo images. Lastly, the limit for how many items to scrape is set here. This was done to prevent large and detailed websites from keeping the scraper occupied for large amounts of time while gathering far more information than useful. All of these settings are simple to adjust allowing it to be maintained and extended with minimal upkeep.

The start-up file exists to allow the scrape to be run without requiring it to be run from the command line. It contains a class that is able to run the scraper outside of the Scrapy environment while still being able to parse a variety of optional arguments. These arguments are the company attributes that the scraper should be searching for in addition to the regular expressions. The argument parser has been implemented to be easily modifiable in case the scraper has to be extended to cover more company attributes. The company website argument is the only required argument. All other arguments are optional.

5.5.2 THE SCRAPER HANDLER

The ScraperHandler class performs the tasks of interacting with the scraper and extracting the scraped data from the CSV file. This class is written in PHP and is the connection between the Python and PHP sides of the system. It also stores this extracted information in a way that promotes ease of use for the Verifier class that will be using this data.
The ScraperHandler can run the scraper with the required company website argument and any available company attributes as optional arguments. The method that performs this action extracts the parameters to send to the scraper from a Company object containing the client provided data.

The extraction and storing of the scraped data is also done in the ScraperHandler class. The class provides a method to extract the information from the CSV file created by the scraper after it finishes. Another method handles the formatting and storing of this data. The end result is an array for each company attribute which contains pairs made up of the specific attribute and how many times it was found during the scraping process. These arrays are available to use by the other classes of the system, such as the verifier.

5.6 VERIFICATION

The Verifier class makes use of all the information gathered by the system to determine which instance of a property is the correct one. The final result of the Verifier is a so-called confidence value which represents how certain the system is that the given information is correct.

The company properties the Verifier receives are immediately enhanced. The user provided versions are given an initial weight which is in general the largest weight because the user will generally provide correct data. The scraped data the Verifier receives already has a rough estimate on its correctness based on how many times each version was found on the website. At this point an array is formed with, for example, all the possible company names and their confidence values based on the user data combined with the scraped data.

Next the Verifier uses the created array in combination with the API Handlers. The Verifier uses the API Handlers to double-check the array and further complement it if needed. It does this by querying the APIs with the user properties and storing the result that is given in response. These results are then used to increase the confidence value if the previously mentioned array already contains the found property. Otherwise, the property found in the results is simple added to the array with an initial confidence value. The setting of the confidence value is based on the number of handlers being used in order to never have confidence values higher than 100, and to prevent the use of many APIs from completely overshadowing the confidence increase provided by the user. After all results have been evaluated the final confidence values are assigned to the corresponding Company object. The system can now use these confidence values to display how likely it is that
the information is correct.
6 QUALITY ASSURANCE

In this chapter an overview is given of the measures that were taken to ensure a high level quality. The first section gives an overview of the tools used to ensure high process quality. The second section gives an overview of the testing techniques used during development. The last section is a combination of the SIG feedback received and the improvements made according to the feedback.

6.1 PROCESS QUALITY

In this section the use of the chosen development framework and version control system are discussed.

6.1.1 AGILE

For the project we applied the agile software development framework, more specifically: Scrum. Scrum involves having weekly meetings to determine the tasks for the week, a backlog to keep track of incomplete and completed tasks, and daily meetings and tasks to keep the team working efficiently. By focusing on the goal to be reached Scrum allowed us to adapt quickly to changing requirements and develop our product in a holistic manner. CHAINels provided us access to JIRA, a project management tool. Here we could list all our tasks for the project and divide them in weekly sprints in order to track progress. In the end JIRA was only used to track the tasks, because the whole group worked with each other daily at the faculty and it was known what each member was doing and what still needed to be done.

6.1.2 VERSION CONTROL

During the project SVN was used as source code management. By using SVN we could work more independently from each other, track code changes and rollback to an early point in time when mistakes were made. CHAINels made a separate branch for us to work on, in order to not interfere with the master branch. After the project and code inspection CHAINels can merge our branch with the master branch.

6.2 TESTING

During the project a large amount of time was spent on creating and running tests. In this chapter an overview is given about the different testing
techniques we have used.

6.2.1 UNIT TESTING

We used unit tests to ensure that each component functioned according to specifications. Unit tests were defined for almost every class. We tested both normal cases as well as special cases (e.g. empty data fields). This allowed us to be almost completely sure that if a component stopped functioning once integrated then the problem would be in the integration and not the individual component.

6.2.2 PERFORMANCE TESTING

Performance testing was also used to see how long the entire process as well as each component of the process would take. For this we made use of self-made and real client data. This self-made data was useful as a control group and to see the optimal performance of the system. The client data, provided to us by CHAINels, allowed us to measure the real world performance of the system. This data consist of a file containing 60 - 200 companies with different amounts of information housed in variety of structures. We also used this data during the development process to see how components performed on it.

RESULTS TUSSEN DE GEVELTJES  The largest CSV file we had was file consisting of 202 companies and 29 data fields for each company. Within these data fields, all properties except state and country were given. Using the three different API Handlers separately, the results in Figure 10 were acquired for the classification. The average run-time is based on ten consecutive runs. The best results are acquired using OpenKVK, having an average run-time of 2.206 seconds and finding all the possible properties. On the other hand, Google Maps is twice as slow as OpenStreetMap, but is 100% accurate whereas OpenStreetMap is only 70% accurate. Using both OpenKVK and OpenStreetMap together, the accuracy becomes 100%, but because the system has to process less data when classifying more and more properties, the run-time is on average almost the same as only running OpenStreetMap.

Figure 11 shows the results for the verification of ten companies. Once again, OpenKVK performs the best whereas Google Maps is still the slowest and OpenStreetMap the least accurate. Although 50% is still not as bad, OpenStreetMap is not able to show suggestions for company names which
is a huge drawback. The total time of the system is close to the sum of classification and verification run-times.

6.3 SIG

The Software Improvement Group (SIG) is a company that is specialised in software quality. They give an in-depth analysis of our code as well as recommendations on how the code can be improve. First a summary of their analysis will be given, followed by the actions taken based on their recommendations.

6.3.1 FEEDBACK

For the first feedback opportunity the code scored almost 4 out of 5 stars based on their maintainability model. The highest score was not achieved due to a lower score for Component Balance and Unit Size.

To score the Component Balance, SIG looked at the partition of the code over the top-level components. In our case the top-level components are 'external' and 'extraction'. SIG’s remark was that it is unclear why this separation was made. Also within these components, there is no subdivision

![API Handlers Accuracy vs Time](image)

**Figure 10:** The performance results for Tussen de Geveltjes CSV File (Classification)
of the code. The recommendation was to divide the code in components with clear naming convention to make future maintenance easier.

To score the Unit Size, SIG looked at the percentage of code that is above average long. Methods that are big can be divided in smaller methods to increase the understandability, testability and maintainability. Within our code, SIG found methods that are relative big and could be broken down in smaller parts. The recommendation was therefore to critically review longer methods to see if they were eligible to be broken down.

The second feedback was, again, almost 4 out of 5 stars. This means that even though the size of the system had increased by 73% we still managed to keep a high level of maintainability. By providing SIG with documentation explaining how our system is part of a much large system it was much clear to them why we choose for our top level components. Even though we have fixed the long methods based on the first feedback certain new methods once again became very long. From their observations SIG concludes that we took their previous advice into account when continuing the development of the system. They are also glad to see that the increase in the system went paired with a similar increase in test-code.

Figure 11: The performance results for Tussen de Geveltjes CSV File (Verification)
6.3.2 IMPROVEMENTS

The following section will elaborate on the improvements that we made to the code after receiving SIG’s feedback. First we will discuss the improvements based on the first feedback. Then the final changes based on SIG’s second feedback.

We received a good score for the first SIG feedback. We had to improve Component Balance and Unit Size. The Component Balance remark involved the fact that the entire system is contained into two top-level components: 'external' and 'extraction'. We were not able to send the rest of CHAINels’ system to SIG due to privacy reasons and forgot to mention that our code is part of a larger system. The other Component Balance remark was related to the fact that inside the two top-level components there is no further component division. We decided to rework the entire component structure. The top-level of our system is in the component 'Create', which houses the components 'external' and 'processes'. Inside the 'external’ component we added the 'api' and 'scraper' components to increase Component Balance. The 'processes' components contains, among others, the Classifier, Entity Recognizer, and Verifier. This improves the ability of future developers of the system to understand and add it. For Unit Size we had several methods that were too long or had functionality that should be in a separate method. We analysed our code and shortened our average Unit Size by transplanting pieces of functionality into their own methods. These changes prevent our Unit Size from making the methods and their functionality difficult to analyze.

For the second feedback we received the same score as for the first. However, this time the Component Balance did not need to be changed. For the Unit Size we went through our code and reduced the size of large methods by transplanting functionality to their own methods. By doing this our code is much easier to understand for future developers.
7 EVALUATION OF THE REQUIREMENTS

In this section we will evaluate the requirements as mentioned in section 3. We managed to successfully full fill almost all of the requirements that we set for ourselves at the beginning of the project. We even manged to include some functionality that we considered a nice bonus. Overall, the project was a great success considering the requirements list was healthy and the final product has lived up to most of the expectations. Since many of the requirements were implemented as intended they are omitted from this section. Instead the focus is put on the requirements that were not met. Unmet and changed requirements are discussed below.

7.0.3 FUNCTIONAL REQUIREMENTS

- Should use SSL (Semi-supervised Learning) techniques to learn from new data - We were unable to implement SSL techniques. This was due to multiple reasons. Firstly, we underestimated the time it would take to implement this feature. Second, the system was performing well without this feature. Lastly, due to limited development time we decided our efforts could be better spend elsewhere.

- Should save the modified file for use as training data - This requirement was not met since it is part of the SSL techniques whose implementation we decided to forgo.

- Could scrape for company logo otherwise should provide a temporary logo - During the development process CHAINels decided that being able to obtain a company logo through our system was of much greater priority then initially communicated. Therefore, we decided to implement this feature with the time freed up by the decision to not implement SSL techniques. The system is able to look for images that are likely to be the company logo through a simple algorithm that checks the name of the image. It is fairly successful, but could be improved by adding methods which attempt to recognize the image itself rather than just the file name.

7.0.4 NONFUNCTIONAL REQUIREMENTS

- Should be extendible to display different languages using CHAINels’ already existing language system - Due to a lack of time we were unable to implement this ourselves. Currently, the CHAINels system itself can run in English and Dutch, but their user base is still purely Dutch.
• **Time between user uploading data and displaying the verified and found data must be less than five minutes** - Due to the potential for very large data sets always staying under this limit is not possible. In order to deal with the long waiting time that can occur we decided to make the verification process asynchronous. This means that the verification component is able to verify a company and immediately display it once done. The rest of the companies continue to be verified in the background. As soon as each company is done it can be displayed to the user. This allows a user to progress step by step through the companies without having to wait for the entire process to finish first.

• **Scraping information per company must be done within five seconds** - This time constraint is also difficult to stay below due to the large variety in website sizes. Most websites stay under this time, however, occasionally a large website can take more time. We implemented a limit on how many pages the scraper will visit, but this limit had to be kept high enough in order to allow the scraper to find enough useful information. The scraper still finds all the necessary information far faster than a human would.
8 RECOMMENDATIONS

• CHAINels could do preliminary research about future business groups. This could lead to improvements in the system. For example, all companies could have the same street name and/or city name. If this happens to be the case, this would mean that these properties do not have to be verified by the system for every company resulting in less resource usage.

• CHAINels should perform some user testing. The system will also be used by future customers. Therefore, it is useful that the system is tested by users to find enhancements in the graphical user interface.

• Although the system is built with a strong focus on companies from The Netherlands, it also works for companies from other countries. However, CHAINels should do research whether there are better external resources to verify foreign companies.
9 CONCLUSION

Given the problems CHAINels had with manually creating company profiles for new customers, we were given the task to find solutions to effectively automate this creation process. Given that clients provide a CSV file with data about the companies that are members of their business group, this has lead to finding solutions for the extraction, classification and verification of the data in these files. For extraction, this means creating a two-dimensional array that represents the data in a CSV file. As for classification and verification, the solutions consist of using external resources such as, but not limited to, Google Maps, OpenKVK, OpenStreetMap or a company’s own website to gather information to be used during these processes. But just as important, an intuitive user interface had to be created for the user to interact with in case some manual verification is needed to help the system.

On the implementation side of the problem, the system has been programmed in such a way that it can be easily maintained and extended with other external sources. Furthermore, to assure the system works as expected, it has gone through thorough unit and performance tests.

During our time working on the bachelor project we made use of our previous experiences and education to successfully complete the project. This bachelor project was an opportunity to work in a real world scenario and produce results that not only satisfy the educational goals, but also satisfy the customers goals. The group worked well together as a whole and the few issues that presented itself were easily resolved. There were times when the goals and deadlines that we set seemed too steep, but through proper planning and communication we managed to successfully complete almost all of the requirements. The techniques and methods that we learned from our bachelor course often came in useful when determining how to tackle each part of the problem. Especially in the early stages the theory learned during the bachelor courses allowed to to hit the ground running. There were also plenty of times when we were unsure on how to proceed effectively, which lead to periods of low productivity while we discussed what was next. These phases were frustrating, but when we managed to get past them it was a rewarding feeling. Learning how to deal with changing customer demands and real world problems is something that previous projects rarely touch on. The experience gained from the bachelor project is therefore one of the most valuable experiences during the bachelor. It allows the student to see that what they have learned in the past three years can be applied to solve problems that might otherwise have seemed unfeasible.
Appendices
A PROBLEM DESCRIPTION

Momenteel wordt het platform CHAINels door verschillende Winkelgebieden, Bedrijventerreinen en Businessclubs gebruikt. CHAINels faciliteert door middel van een website en een app de interne communicatie van bedrijfs- groepen. Het bachelorproject zal zich richten op de aanmeldprocedure van een community. De huidige situatie is dat een groep een ledenbestand aanlevert waaruit wij handmatig accounts genereren. Deze profielen zijn dan, naast wat basis informatie (bedrijfsnaam, email), vrij kaal en incompleet. Wij willen dit proces meer automatiseren en ook uitbreiden. Hier ligt dan ook de uitdaging voor jullie: Automatiseer het start proces van een community doormiddel van beschikbare bronnen, zowel aangeleverde bronnen als online bronnen. Zo zijn er vaak bedrijven/ledenlijsten (CSV), bedrijfswbsites en social media met benodigde gegevens beschikbaar. Door slim gebruik te maken van deze bronnen zou een community direct automatisch opgezet kunnen worden. Een paar kernproblemen zijn hier dus het omgaan met incomplete, onvolledige en ongestructureerde data, en het crawl van websites en social media (data mining) om zo voor alle bedrijven een zo compleet mogelijk profiel te genereren. Verder is er veel ruimte voor eigen inbreng en kunnen prototypes snel in praktijk getest worden. CHAINels is een online platform ontwikkeld in (object georienteerd) PHP, JavaScript, HTML, CSS met daarachter een (NoSQL) Redis database. Kom vrijblijvend een keer langs dan vertellen we je meer en kan je waarschijnlijk je bachelor succesvol afronden! (De afgelopen twee jaar zijn er al 4 bachelor projecten succesvol bij ons afgerond met een 8+)

Company description
CHAINels is een startup in YES!Delft, de incubator van de TU Delft. Een team van 6 man met een achtergrond aan de TU Delft (Technische Informati ca, Industrieel Ontwerpen en Technische Bestuurskunde) is overtuigd dat ondernemers in een winkelgebieden of bedrijventerreinen te allen tijde met minimale inspanning moeten kunnen profiteren van het collectief waarvan zij onderdeel zijn. Het bedrijf heeft begin 2013 een investering gekregen en inmiddels gebruiken duizenden bedrijven het platform.

Auxiliary information
CHAINels B.V. Molengraafsingel 12 2629JD Delft 015-3642667 info@chainels.com
This project is also associated with: Vincent Koeman koeman@chainels.com
B ORIENTATION REPORT
Project Extraction - Orientation Report

Pedram Ardeshirzadeh (4109252)  Kevin Kessels (4078586)  
Ping Wan (4111443)  

June 18, 2014
Contents

1 Introduction 3

2 CHAINels' System 4
  2.1 Programming Languages 4
  2.1.1 HTML5 4
  2.1.2 CSS 4
  2.1.3 Javascript 4
  2.1.4 PHP 5
  2.2 Model-View-Controller 5
  2.2.1 The Company Model 5
  2.3 PHO 6
  2.4 Integration 6

3 Data Handling 7
  3.1 Data Example 8
  3.2 Named Entity Recognition 8
  3.3 Regular Expressions 10
  3.4 String Similarity 10

4 Scraping and Verifying the Data 10
  4.1 Scraping the Information 10
  4.2 Verification of Information 11
  4.3 Existing Scrapers 11
  4.3.1 Scrapy 12
  4.4 Beautiful Soup 12

5 Conclusion 13

References 13
1 Introduction

CHAINels is a startup providing business groups and its members an online platform to effectively communicate with each other in order to collaborate better and faster. These business groups can consist of many different companies. From well known companies such as Heineken to small startups like CHAINels. As new groups become clients they have to be entered into the system. Currently this is done manually by the employees at CHAINels. This means finding and verifying data for hundreds of companies. Due to CHAINels being a small company, spending time and manpower to manually setup clients ties up a large amount of their valuable resources. By largely automating the setup process these resources can be refocused.

To implement this autonomous feature it is of importance to highlight the two main problems: 1) the understanding of and integration with CHAINels’ existing system and 2) handling the data clients provide. For the first problem, we will take a look at how CHAINels’ current system is implemented and functions. By understanding the programming languages and components that are used it becomes easier to come up with a solution that can be integrated with CHAINels’ system. After understanding CHAINels’ system, the integration will be the next step that will be discussed.

The second problem can be separated into several smaller sub-problems and the process of finding solutions for all these sub-problems will henceforth be called the data-pipeline. The first step in the data-pipeline is receiving the data. Clients can provide data files with their current administrations. It is important to take into consideration that there is no general guideline and every client stores their data differently. The second step is recognition of the data. As can be seen later on in this report, not every data file provides enough information or the same format of information. Within this step, the importance lies in categorizing and understanding which field, for example, is the address field and which one is the name of a company. The next step is verifying the categorized information. It is possible for data to be outdated or be incorrect due to some typing error. It is therefore important to verify the correctness of the data. This can either be done by asking the client whether every piece of information is correct or by automatically searching the web for the right information through the use of a scraper. Social media for example can be used to find the data for verification. The final step of the data-pipeline consists of filling in the missing information. Just like with the previous step, finding missing information can either be done with the use of a scraper or asking the client.

In Chapter 2 of this report, CHAINels’ current system will be discussed.
This will include which programming languages are used as well as describing their software architectural pattern. Chapter 3 discusses the first two steps of the data-pipeline. Subjects such as Named-Entity Recognition and Regular Expressions will be discussed to explain the categorizing of the data. The final two steps of the data-pipeline will be discussed in Chapter 4. Here the benefits and limitations of using scrapers will be discussed as well as different scrapers in existence. The verification process will also be explained in detail. In Chapter 5 we will provide our final solution for implementing the previously mentioned autonomous feature.

2 CHAINels’ System

CHAINels is currently active for more than two years. It therefore already has a fully functional system. In this section an overview of this system with its programming languages and frameworks will be given.

2.1 Programming Languages

The following Sections will describe the programming languages CHAINels is using in their current system.

2.1.1 HTML5

HyperText Markup Language (HTML) is a markup language used to create websites with HTML5 being the fifth revision. A website consists of multiple HTML elements that start with a begin tag (<tag>) and end with an end tag (< /tag>). The webbrowser can interpret these tags in order to render a more appealing website. HTML(5) is a standard for building websites.

2.1.2 CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language like HTML. With CSS there is good separation between the page content and the look and feel of it. CSS is also a standard in building websites.

2.1.3 Javascript

Javascript is a dynamic programming language. It can be used either client-side or server-side. CHAINels uses javascript primarily client-side for han-
2.1.4 PHP

PHP Hypertext Preprocessor (PHP) is the main programming language of CHAINels and is used the most in their system. PHP is a server-side programming language that is processed by the webserver in order to send HTML pages to the browser. CHAINels uses PHP because it supports the powerful programming paradigm Object-oriented programming (OOP) that CHAINels also incorporates as much as possible.

2.2 Model-View-Controller

The CHAINels system is built using the software architectural pattern Model-view-controller (MVC). As the name suggests MVC divides a system in three parts: the model, the view and the controller.

The model represents the knowledge of the system. It contains all objects that are stored as well as the relationship between different objects. The view consist of all user interface elements that are displayed to the users. The controller reacts to events from the view and can also update both the model or the view.

While applying the MVC architectural pattern, CHAINels tries to translate every component of the MVC into classes. For example each view, such as a button or a menu, is a class that creates the markup for the resulting page. Each controller also has its own class. By doing this CHAINels can fully utilize the strengths of OOP.

2.2.1 The Company Model

The most important model that we have to work with is the company model. Each company entity has the following:

- A company name

- An about object, which holds information such as a small description of the company, business number and business hours.

- An address object, which holds information such as street name, street number and zipcode.
• An industry key and industry label, which we will ignore because clients will manually select available choices at a later stage during the profile creation.

• A contact object, which holds the email, phone number and website address.

• A company logo, either a large or small picture.

There are also some other properties, but these are not relevant for us as they are handled automatically by the backend.

![Company Class Diagram]

Figure 1: The company class

2.3 PHO

A notable library that CHAINels uses is PHP HTML Objects (PHO). PHO is a public project that creates HTML objects using PHP. This reduces the clutter that occurs when using HTML and PHP interchangeably.

2.4 Integration

Our feature will be a separate entity in the CHAINels’ system. Therefore integration will be straightforward. The feature will need a controller and a view. After processing the data, as discussed in Sections 3 and 4, we
will use the company model in order to create company entities that will be added to the database after verification by the user.

3 Data Handling

The first two steps of the data-pipeline consist of receiving and recognizing the data. First, we will provide some example data files and discuss the important aspects of these files. This includes discussing the different file types, but also the variations within the files such as the usage of abbreviations or different ways of providing a phone number. After that we will describe what techniques can be used to recognize and extract the right data regardless of variations that exist in every file. The methods Named-Entity Recognition, Regular Expressions and String Similarity will be discussed for this purpose.
3.1 Data Example

Not all clients provide the same type of data file, but in most cases the file is either an Excel file or a CSV file. We will first take a look at a few example Excel files. As seen in Figure 2, the columns in the Excel file are the different properties of each company and the rows are the company entities. The first thing to note is that while in Figure 2 the first row is used to specify what property each column holds, the same does not hold for Figure 3. Therefore, it is important to have a method that categorizes each column if it is not already done so in the file. As can be seen, the files do not contain the same data. While Figure 2 provides information about the contact person and the region of the companies, Figure 3 does not. Inconsistencies also exist within the same file. As Figure 4 shows, not all postal codes and phone numbers follow the same format. Fields could also be empty resulting in missing data, but more on that in Section 4. It is also possible that the file contains old information. For example the email address provided is no longer in use or abbreviations are used in addresses.

3.2 Named Entity Recognition

Named Entity Recognition and Classification (NERC) or simply Named Entity Recognition (NER) has been around for over a decade now (Nadeau & Sekine, 2007). NER is a subtask of information extraction that locates and classifies elements in text into pre-defined categories. Throughout the years, the systems for recognizing previously unknown entities have shifted from being based on handcrafted rules to supervised machine learning. As Baluja, Mittal, and Sukthankar (2000) mention, systems based on handcrafted rules require large amounts of time and expertise to develop and are not easily portable to new languages and genres. Supervised Learning on the other hand is based on the idea of having large collections of training data to study the features of example named entities and come up with rules to recognize new named entities. For SL to work though, there is the requirement of having such a large collection of example data. Due to this shortcoming, Nadeau and Sekine (2007) discusses two alternative methods: Semi-supervised Learning (SSL) and Unsupervised Learning (UL). With SSL the user provides a small sample data and the system starts the learning process with that, learning more and more over time. UL is more based on finding clustered groups and recognizing named entities in these groups.

NER is useful when clients do not provide information about the fields as mentioned in Section 3.1, but it has its limitations. As mentioned before,
the data files received only have fields that lack grammar. This means that the prominent feature of NER systems, using the context around a word, can not be used. In order to deal with this other kind of features need to be found. For example, if the companies of a business group are (mostly) located close to each other this would mean the city they are located at will be mention almost in every row. As another example, companies in The Netherlands can have either N.V. or B.V. in their name. This would mean that a column with some amount of rows having this feature is probably the column with the company name. Besides finding a decent amount of these kind of features to use, another problem is whether the system should be supervised or not and if so, how supervised should the system actually be?

The benefits and limitations of the different systems have been discussed previously, so the question that remains is which one will work the best in our case? Not going for supervised is usually done when the system tackles one specific genre or language. If the knowledge about that certain genre or language is available, it is easier to build and provides more accurate results. In our case, it is hard to argue we have the knowledge about all the possible formats clients provide their data. If CHAINels also decides to go across the borders, it would become a problem since only The Netherlands will be supported with no scalability. This leads to a supervised system. SL requires a large amount of training data which is not available, leaving SSL and UL as the only options. SSL is the best of two worlds where provided a small amount of data, the system is able to learn on its own. This is most useful in our case where there is a general knowledge of how all the data is formatted, but to actually handle all the possible formats, it is best to let the system learn on its own. While UL could be used for recognizing postal codes and city names, as they appear the most in data files, it cannot be used to deal with all of the entities.
3.3 Regular Expressions

Regular expressions are sequences of characters that form a search pattern. An algorithm can scan a text and see if a certain string matches the pattern. In the data that is provided by the companies certain properties can be recognized using regular expressions. For example, postal codes and phone numbers follow a certain pattern and have constraints on the way they are defined. This results in a finite number of variations. All these variations can be covered by regular expressions (Clarke & Cormack, 1997).

3.4 String Similarity

String Similarity is a string metric that measures the similarity between two strings and provides a number called the distance between the two strings. The distance is algorithm-specific, meaning an acquired distance with algorithm A is not necessarily the same with algorithm B. One of the most well-known string metrics is called Levenshtein Distance. Levenshtein Distance is the distance between two strings, describing the number of character edits needed to transform the first string into the second string. String Similarity is used for purposes such as fraud detection and spell checking. With String Similarity, we are able to find incorrect data in the given files. This is done by comparing for example a set of addresses with each other in order to find whether one of the addresses is misspelled. Another possibility is comparing strings with data gathered through scraping (Section 4) in order to confirm the company properties contain the right information.

4 Scraping and Verifying the Data

In this section a short overview of what scraping is will be provided. Furthermore an explanation is given of how scraping will be utilized to gather and verify information. Lastly, two existing scrapers will be compared.

4.1 Scraping the Information

Web scraping is the act of automatically extracting information from websites by simulating the actions of a human user accessing the page through a browser. Through this simulation web scraping is able to obtain unstructured data and convert it into structured data that can be analyzed automatically. By scraping the websites of companies that are about to join the CHAINels network we will be able to cross reference this data with the
client provided information in order to determine whether the information is correct and up-to-date. Through the scraping process we will also be able to obtain missing information that the client’s did not provide but is available on websites such as LinkedIn and individual company websites.

4.2 Verification of Information

The profile creation process at CHAINels begins with the client providing some form of their group’s administration. As discussed in 3.1, the provided data often contains old information or the same information is written in varying ways. When information is incorrect the company does not benefit nearly as much as it could from being a part of CHAINels. Similarly when an address in written in multiple formats manual intervention is currently required to determine whether two streets names represent the same street or are actually different streets. Through the use of the scraped data we can solve these problems automatically. In the case of verification we already have a client provided version of the information. By comparing the client provided version and the scraped versions with each other we can draw an intelligent conclusion on whether or not the information is accurate. For instance, if we find the address of a company five times through scraping and they are all the same as each other and the same as the client provided one then we can be sure that this information is correct. There will be situations when we will find that this is not the case, then we can mark the company property as unsure and use the two most found options as suggestions for the client when requesting them to verify the property. The automatic collection and verification of information together with the suggestion feature will make it easy for the client to setup their group’s profiles without CHAINels having to manually perform these tasks. Our feature will also make it possible to create a profile rapidly creating an opportunity for the client to trial the system quickly.

4.3 Existing Scrapers

After researching available scrapers we have narrowed our interest to two candidates: Scrapy and Beautiful Soup. Scrapy focuses on simplicity and extensibility. Beautiful Soup has a short setup time combined with useful features such as the ability to trade speed for flexibility by switching parsers. In the next two sections an overview of each candidate will be given together with their pros and cons.
4.3.1 Scrapy

Scrapy is a fast high-level web scraping framework used to extract data from Web pages. It provides techniques for extracting information from Web pages that don’t have any API or mechanism to facilitate this process. For CHAINels’ clients the lack of such a mechanism will often be the case since setting this up for a small company website is generally not beneficial. Another benefit of Scrapy is that it has a simple yet extensible design. This means that less time will have to be dedicated to learning and implementing the scraper, while still being ready for any future improvements or additions that might need to take place. It also has build-in support for cleaning and sanitizing the scraped data which will allow for an easier validation process. CHAINels has mentioned that, if possible, it would like to have the company logos obtained through our feature. Scrapy provides a media pipeline function that enables the downloading of images associated with any scraped information. Through the use of this pipeline implementing the finding of company logos might be possible depending on the available content. Through these features Scrapy fullfills all the requirements that we have for our scraper while also providing some extras such as the media pipeline.

4.4 Beautiful Soup

Beautiful Soup is a Python library designed for web scraping. It also has the ability to obtain information from websites that don’t provide an automated mechanism. One of the main advantages of Beautiful Soup is that it is very fast to setup using a provided toolkit for dissecting a document and extracting the desired information. Through the use of this toolkit we can save time implementing the scraper, which leaves more time for developing the other parts of our project. Beautiful Soup also handles encoding issues automatically by converting any incoming documents to Unicode and any outgoing documents to UTF-8. Since Beautiful Soup is a Python library it can make use of any of the existing Python parsers. These are plentiful and each of them has its advantages and disadvantages. A parser can be chosen based on which aspect of the scraper needs improvement without having to rework the scraper itself completely. Beautiful Soup will allow rapid deployment of a scraper that will fullfill the basics needs of our scraper, but it is limited in terms of extendability.
5 Conclusion

CHAINels’ current system is built with the Object-oriented programming paradigm and Model-View-Controller architecture. Our feature will therefore also be programmed with these two patterns in order to easily integrate with CHAINels’ system. The feature will be its own separate system resulting in a minimal integration process.

Given the provided data files, it is important to classify and extract the correct data. This allows the other parts of the system such as the verification of the information and the scraping to perform better. This will be done with the use of Named Entity Recognition and Regular Expressions. Semi-supervised Learning will be the best recognition technique to use in our case, as it provides the right amount of supervision and machine learning given the provided data files.

The scraping of information is a key component of our feature and ensuring the best possible scraping results will allow the verification process to be as accurate as possible. After having compared the two most appropriate scrapers available to us, Scrapy provides all the features that we need while remaining simple to use and future-proof through its extensibility. The faster setup time for the basics that Beautiful Soup provides cannot make up for the complete package that Scrapy offers.

References


C PLAN OF APPROACH
Project Extraction - Plan of Approach

Pedram Ardeshirzadeh (4109252) Kevin Kessels (4078586)
Ping Wan (4111443)

June 18, 2014

Contents

1 Introduction 3
  1.1 The Company ........................................ 3
  1.2 Structure of The Document ............................ 3

2 Project Assignment 3
  2.1 Project Environment .................................. 3
  2.2 Project Goal ............................................. 4
  2.3 Assignment Description ............................... 4
  2.4 Deliverables .......................................... 4
  2.5 Requirements and Limitations ....................... 5
  2.6 Conditions ............................................. 5

3 Approach and Planning 5

4 Project Design 6
  4.1 Stakeholders .......................................... 6
  4.2 Personnel ............................................. 6
  4.3 Administrative procedures ........................... 6
  4.4 Reporting ............................................. 7
  4.5 Resources ............................................. 7

5 Quality Assurance 7
Foreword

This is the Plan of Approach for our bachelor project at the company CHAINels. In this document we will discuss how we will approach the project to complete the given assignment from CHAINels, alongside a brief description of what the company expects from us as well as the planning.

Summary

During this project, an online solution will be given for CHAINels to automate the initial profile creation of companies. For the duration of approximately ten weeks, we will work on finding and implementing a solution to provide an as good as possible end product. We will work closely with the employees of CHAINels to make sure the goals of the project align with the product requirements.
1 Introduction

During the end phase of the Computer Science bachelor students need to work on a project for a company. Students have the option to find a company themselves or choose one of the companies that already have an agreement with TU Delft to provide a bachelor project. For our project we have chosen for the company CHAINels after contacting them and getting a better understanding of the assignment. We chose CHAINels because the assignment they provided was met with interest by all of the group members.

For this project our group consists of the following three Computer Science students: Pedram Ardeshirzadeh, Kevin Kessels and Ping Wan. With this Plan of Approach we take the first step towards clarifying what we are going to do in the given period and how we are going to do it.

1.1 The Company

CHAINels is a startup operating in Delft. Their main service is to help companies that are in the same business group by providing them with a tool to collaborate better and faster in the field of knowledge, network, safety, collective purchasing and policy. With this tool the companies will be able to create a profile on the CHAINels website where they can communicate with other companies within the same group or read the bulletin with important news or alerts.

1.2 Structure of The Document

In Chapter 2 we will describe the project assignment in more detail. We will discuss the environment of the project as well as its goal, deliverables and requirements. In Chapter 3 a planning of the project period is given and a description of the collaboration method. Chapter 4 will discuss the project design, explaining which people are involved, what they do, which tools they use and how they report their progress. Finally, in Chapter 5 the quality assurance of the project will be discussed.

2 Project Assignment

In this chapter we will fully describe and clarify the project assignment. First we will describe the given problem alongside the project goal, assignment and the conditions for a successful product. Then we will discuss the product and its requirements and limitations.

2.1 Project Environment

CHAINels is a company that provides an online platform for business groups to effectively communicate with their members. As new groups become clients
they have to be entered into the system. These groups are often already established, meaning that large amounts of data have to be manually entered into the system. This is done by providing CHAINels with the administration data the group already possesses, usually either an Excel or CSV file, and CHAINels creating every aspect of the profiles manually. However, the information provided often contains incorrect or missing information. This currently requires manual verification of the information and contacting the client for the missing information.

2.2 Project Goal

CHAINels is currently a small company. This means that spending time and manpower to fully setup new clients ties up a large amount of their valuable resources. By largely automating the setup process these resources can be re-focused. This means not only automating the input of the client information, but also the verification and collection of missing information. By handling this CHAINels will only be minimally involved in the profile creation process whereas the client will be more involved, but with less difficulty. The automation will also reduce the time between client signup and having access to the service. Through our efforts to improve the startup phase CHAINels wants to reduce their resource investment and make the process as effortless as possible for the client.

2.3 Assignment Description

Our task is to create a program that will aid CHAINels in integrating a new client with their system by automating the information extraction and profile setup. Preliminary verification will be done automatically, but will require manual input to some degree. The program will also attempt to find the missing information and search for other sources with the same information to verify their accuracy. The process will not become fully automated since it is not possible to be completely accurate. This is because of the ambiguity of the information’s structure, and the possible incorrectness of the information. The program will provide an appropriate and efficient way to process any manual verification.

2.4 Deliverables

We will deliver the program together with its code and documentation. This will allow CHAINels to fully understand and expand the program as necessary. We will also provide test code with documentation for verification of the system’s functionality. Furthermore, our Plan of Approach and the Orientation Report will be provided upon completion in the first two weeks.
2.5 Requirements and Limitations

Must

- Create as much of a new client’s profile as possible with the given information.
- Give an overview of detected missing and incorrect information.
  - Provide an appropriate way to manually resolve the above.
- Scale to handle large groups as well.
- Integrate into CHAINels’ system.

Should

- Detect missing information 100% of the time.
- Attempt to verify correctness of the given information through crawling.

Could

- Attempt to find missing information through crawling.
- Attempt to verify correctness of the found information through crawling.

Won’t

- Be autonomous.
- Detect all incorrect information.

2.6 Conditions

To be able to deliver a viable product certain conditions need to be met. First off, team members need to have appropriate knowledge of the programming languages that will be used in this project. This will be mostly PHP for usage of and integration with CHAINels’ system. Team members also must be familiar with the system and its components. The requirements for the desired solution should be known before we start the implementation.

3 Approach and Planning

The project has a minimum duration of ten weeks. The following is the planning for these ten weeks:

- In the first week a plan of approach will be made.
• In the first and second week intensive research will be performed. In this stage the most suited techniques and tools will be examined and chosen.

• In week three and four the essential requirements will be gathered and the design of the system will be made.

• Week four to week eight will be planned for the implementation phase. During these weeks we will write the code for the system. Throughout the implementation the code will be frequently tested.

• In week nine and ten a large amount of time will be spent on testing. The finishing touches will be added and the documents will be completed. Time will also be spent to create and practice the final presentation.

The project will be performed using the agile development framework SCRUM. This means the project will be divided in multiple sprints. In each sprint new functionalities will be added to the system. There will be a product backlog containing ordered requirements that need to be completed in order to deliver a viable product. In each sprint important requirements from the backlog will be completed.

4 Project Design

4.1 Stakeholders
The stakeholders in this project are:

• Team members: Pedram Ardeshirzadeh, Kevin Kessels and Ping Wan.

• Main contact at CHAINels: Vincent Koeman.

• Business support at CHAINels: Erwin Buckers, Willem Buijs, Sander Verseput.

• ICT support at CHAINels: Christiaan Bolivar.

• Project coach: Assistant Professor Alessandro Bozzon.

4.2 Personnel
The team members will work 40 hours per week on the project during the span of approximately ten weeks while having a meeting once a week with the main contact at CHAINels and every two weeks with the project coach.

4.3 Administrative procedures
Several tools to monitor the project, as recommended and also used by CHAINels, will be adopted:
• JIRA, a web-based project managing tool. It supports the SCRUM development framework which we will be using.

• Google Drive will be used to store important documents.

• SmartSVN, a client for SVN will be used for code management.

• Confluence, a web-based team collaboration tool used for managing meetings.

• Jenkins, an open source continuous integration tool used for testing purposes.

4.4 Reporting
During the weekly meetings with CHAINels the team will report to the main contact person at CHAINels. Every two weeks, the team will also report to the project coach. During the week where there’s no meeting with the project coach, a brief report of our activities will be e-mailed to the project coach.

4.5 Resources
The team will use their own laptops and PCs and try to reserve a meeting room at EWI. CHAINels has also given access to their SVN repository.

5 Quality Assurance
To maintain a high level of quality different measures will be taken. First off, it is important that we regularly meet each week to work together and discuss the progress of the project. Each week we also have a meeting with the client in order to find out if our solution is still in line with their problem. The code will be well documented, commented and tested with (unit) tests throughout the whole project. In addition to our own test the code will also be sent to the software improvement group (SIG) for testing. Furthermore, we will use a version control system to prevent data loss and to have the ability to roll back to an older version in case something went wrong.
D SIG FEEDBACK

D.0.5 FIRST FEEDBACK

De code van het systeem scoort bijna 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Component Balance en Unit Size.

Voor Component Balance wordt er gekeken naar de verdeling van de code over de top-level components, in dit geval 'external' en 'extraction'. Op het eerste gezicht is het niet duidelijk waarom de code opgedeeld is in deze componenten. Daarnaast is binnen deze componenten geen onderverdeling gemaakt van de code, wat het voor toekomstige ontwikkelaars lastiger te analyseren maakt. Het is aan te raden om de code onder te verdelen in componenten met een duidelijke naamgeving om zo toekomstig onderhoud makkelijker te maken.

Voor Unit Size wordt er gekeken naar het percentage code dat bovengemiddeld lang is. Het opsplitsen van dit soort methodes in kleinere stukken zorgt ervoor dat elk onderdeel makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden wordt. Binnen de langere methodes in dit systeem, zoals bijvoorbeeld de 'extractScrapedData'-methode binnen 'Scraper-Handler', zijn aparte stukken functionaliteit te vinden welke ge-refactored kunnen worden naar aparte methodes. Commentaarregels zoals bijvoorbeeld '// Iterate through each line of the CSV file' en '// Filter out any empty elements left in the arrays' zijn een goede indicatie dat er een autonoom stuk functionaliteit te ontdekken is. Het is aan te raden kritisch te kijken naar de langere methodes binnen dit systeem en deze waar mogelijk op te splitsen.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkel fase. De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test-code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

D.0.6 SECOND FEEDBACK

In de tweede upload zien we dat de omvang van het systeem is gestegen (73%), maar dat daarbij de score voor onderhoudbaarheid gelijk is gebleven. Door de meegeleverde documentatie is het duidelijker waarom er gekozen is voor de huidige structuur. Wat betreft de Unit Size zien we een minieme stijging. Wat hier opvalt is dat de eerder genoemde methode is aangepakt,
maar dat er wel andere langere methoden zijn bij gekomen. Als laatste zien we naast een stijging in de productie code ook een stijging van de hoeveelheid test code.

Uit deze observaties kunnen we concluderen dat de aanbevelingen van de vorige evaluatie zijn meegenomen in het ontwikkeltraject. Het is goed om te zien dat naast een stijging van het volume van het systeem ook nog steeds een stijging in het volume van de test-code te zien is.