Using Supervised Machine Learning To Identify Swiss Companies' Websites

Master Thesis

Zeno Bardelli

University of Fribourg

Supervisor
Prof. Dr. Philippe Cudré-Mauroux
eXascale Infolab, Department of Informatics, University of Fribourg

Co-Supervisor
Ines Arous
eXascale Infolab, Department of Informatics, University of Fribourg

Co-Supervisor
Dr. Ljiljana Dolamic, armasuisse

September 2020

Abstract

Finding companies website is a fundamental task for many applications ranging from building a database of business to automatic collection of data directly from the websites. Yet, automatically identifying the website of a company is challenging. Existing methods rely heavily on the first result shown by search engines. However, the first result presented by search engines does not necessarily correspond to the company website. In some cases, the first result is the website of a company with a similar name, in other cases it is a page on a yellow pages website. The solution proposed in this work is to identify a company's website given its properties and the results of Google Search. We have conducted extensive experiments with existing machine learning classifiers on a data set of Swiss companies. We have studied multiple methods to improve the performance of these classifiers, including hyperparameters tuning, features selection and post processing. Experimental results on identifying Swiss companies website show that our approach improves the results obtained by a Google search by 26.60% in terms of F1 score, from 62.58% to 89.18%.

Acknowledgements

First of all I would like to thank the PhD student Ines Arous, for having followed me with constancy and professionalism in carrying out this work, allowing me to keep my focus and always have a clear goal. I would also like to thank Prof. Dr. Philippe Cudré-Mauroux for the great availability and help I received. My thanks also go to Armasuisse and their representative who was directly involved in this project, Ljiljana Dolamic. Without them this work would not exist.

Finally, I would like to thank my family and friends who, with their unconditional support, have allowed me to get this far.

Contents

List of Figures

List of Tables

1 Introduction

A website allows users to discover the company's activity and to promote its new products. Automatically collecting companies' websites is therefore fundamental as it gives access to the company profile and its activity. The problem we are tackling in this master's thesis is to identify, given some specific features, a company's website. Our primarily use case is the one of companies in Switzerland.

Existing techniques are mainly commercial products consisting of a handful of services as softwares and APIs. Using their free trial versions, we tested their performance on a sample from our data set and we report 84% of accuracy.

With this work we aim to achieve better performance. The solution we propose combines search engines with machine learning. The main idea is to analyze the first results of Google Search individually and use supervised learning classifiers to determine which of them are the correct website of the searched company. We then implement four different classifiers and optimize their performance. In addition, we also implement a scraper to retrieve company domains directly from a yellow page site. Our experimental results show that we achieve a precision of 87.20%, a recall of 91.24% and F1 Score of 89.18%.

This project is a collaboration between the University of Fribourg and Armasuisse, the R&D agency of the Swiss Department of Defence. Armasuisse has generously provided us with two data sets: a data set of 703 175 Swiss companies and a data set with the results obtained from Google Search searching companies' names. The properties of the data sets will be discussed more in detail in Chapter 3.

Since this is a highly practice-oriented work, an important part of it was the development of programs written in Python to bring us closer to our goal. These programs are attached in a digital folder being an integral part of this project. The folder structure is explained in Appendix B. We will explain the strategies and techniques used to get the reported results.

The report is structured as follows. We will begin by reporting existing techniques for retrieving company websites and presenting the classifiers chosen in Chapter 2. In Chapter 3, we will describe the data set that served as a starting point, supplied to us by Armasuisse. More precisely, we will discuss the features used for the classification. In Chapter 4, we will explain how we approached the problem and the procedures used to refine the performance of the classifiers. The following chapter, Chapter 5, will report all the results obtained during the master thesis. In Chapter 6, we discuss the results and future work. Finally, Chapter 7 concludes the document, reviewing the most important results obtained and lessons learned.

2
Background

In this chapter, we first present some of the related work from the scientific literature. Then, we report existing commercial products that automatically identify company websites. Finally, we briefly present four classifiers that we implemented for supervised learning, illustrating the advantages that led us to use them.

2.1 Related Work

Our work connects to many domains in Computer Science mainly Entity Linking, Search Engine Ranking and Crowdsourcing. In what follows, we discuss the literature that connects our problem to these domains.

Entity linking consists in associating an entity that appears in an unstructured text to an entry in a Knowledge Base. For example, an entity could be a person, a place or an organization and a Knowledge Base could be Wikipedia [19]. The difficulty in entity linking comes from the fact that, depending on the context, one can refer to different entities, which may share the same name. Milne and Witten [24] proposed a method to identify significant terms in text documents and match them with the corresponding Wikipedia page using machine learning. Dredze et al. [20] have defined a technique specialized in the disambiguation of entities using the content of the Wikipedia page. Among the various features they used, they checked whether the known information about the entity appears on the page, or the similarity between the document where the entity appears and the page. *Recognyze* is an named entity linking component developed by Weichselbraun et al. [26]. It uses sophisticated modular pre-processing techniques to identify ambiguous names, context and structure. *Recognyze* uses knowledge base linked enterprise repositories derived from data sources on heterogeneous enterprises, like online databases. Based on the company data contained in the knowledge base, it identifies the companies in the unstructured text. These techniques inspire our work as we need to link an entity (in our case a company) with the appropriate website.

Search Engine Ranking includes the set of methods to determine where a search result should appear on a result page. The principle is that the best results appears first. Since any search can have potentially thousands of results, it is very important for the user to obtain the most relevant results. The ranking uses various factors and the interest of many sites is to satisfy them as much as possible, so that they appear higher in the results. It is also important for companies to have as high a score as possible in the ranking, as users generally only consider the very first results. Therefore, different techniques have been studied

to improve the position of the site in the ranking of search engines that involve the integration of certain features in the website. As a result, many company sites adhere to certain standards that allow us to better identify them. That process is called Search Engine Optimization. Killoran [21] has in fact indicated some useful techniques to improve the visibility of the site on search engines. According to the study, it is important to include keywords in the webpages's text and title that could be of interest to the audience of the site. It would therefore be useful to have keywords linked to a company's activity, so that it can be verified on candidate sites as company sites, or added as keywords to Google search. Luh et al. [23] estimated Google's search engine ranking function, so that the could determine the most important factors. The study found that PageRank is the most important factor, followed by title, snippet and URL. These can serve as examples to understand that it is important to take into account in the page title and the URL when searching for a company's website.

The last domain is crowdsourcing, i.e. the collective development of a project with the help of an ideally large number of external individuals on a voluntary basis. Crowdsourcing can be used to improve a search engine results. Kim et al. [22] have in fact proposed to use the evaluation of human users of a search engine to identify the best results and exclude less relevant ones in the future. Moradi [25] has analyzed the concept of a search engine based on crowdsourcing, reporting its benefits and proposing best practices. According to Moradi, the search can be improved by considering what the crowd searched for with similar keywords, as common errors in the query are corrected. It is also useful to know how the results are evaluated by users, in order to prioritize the best ones.

2.2 Existing Services

Several have already tried their hand at automating the search for company sites, demonstrating that this type of work has its own niche of appreciators. We have in fact found several products and APIs that perform the task of finding the website of a company from its name. In the following we will briefly present four products.

The first one is an API called simply *Company Name to Domain API* from Clearbit [2]. Its strategy is to find the exact match with the company name and select the site with the highest traffic. It is among the first names to pop up by looking for similar products and is recommended by several users. The *Domain Name Finder* application from Phantombuster [4], a company offering multiple web scrapers, selects a search engine from an unspecified list and retrieves the first result by searching for the company name. There is then Powrbot Inc.'s *Company Data Enrichment* [5], which finds the website but also lots of other information from the company name. No details are given on how the program works. However, the developers claim to address problems using data analytics, automation and machine learning. The principle behind the last program is different. As the name suggests, *Company URL Lookups using Bing Search* by Blockspring [3] uses Microsoft's search engine. It is not known whether the program simply collects the first Bing result or whether it performs other operations. The only thing that requires as input in addition to the company name is its geographical region.

We have therefore provided a perspective on the different products that offer a solution to the problem of automating the search of companies' websites.

2.3 The Classifiers

There are many classifiers in machine learning. Each one has its advantages and is particularly suitable for certain types of data. We have therefore implemented four classifiers for our problem: Random Forest Classifier (RFC), Support Vector Classifier (SVC), Logistic Regression (LR) and Multilayer Perceptron (MLP). For the implementation, we used Scikit-learn [9]. Scikit-learn is an open source machine learning

library for Python. It provides many tools for machine learning, including many classifiers, model fitting, pre processing and others.

The RFC [1] is a classification algorithm in which many decision trees vote to determine the prediction of the model. The main characteristic is that the trees are relatively unrelated, so that the same errors are not made by many. The errors of individual trees are therefore counterbalanced by the majority. Of course, it is important to build the trees so that they are not related, using two techniques are used that differ in creation from a normal decision tree. The first is bagging. Since decision trees are strictly dependent on the available data, when creating a Random Forest tree we use a random subset of the training set for each tree. The second is to select a random subset of features when splitting a node. The RFC is distinguished by good performance and the ability to assess the importance of individual features. However, it is slow in predictions and may require a lot of memory for the creation of many trees. We choose to use RFC because it has the reputation of being one of the best classifiers and can be used to solve a large number of problems.

The principle behind SVC [16] is to consider features as coordinates in an n-dimensional space, where n is the number of features. The aim is to find a hyperplane that separate the data points as good as possible. The label classes are then divided by such hyperplane. To train the classifier it is necessary to maximize the distance between the points and the hyperplane. An important requirement for training the classifier is to standardise the data, which is why we used the StandardScaler [15] function from Scikit-learn to remove the mean and scale to unit variance. The SVC does not scale well with large amounts of data and features. In fact with such conditions the memory usage increases and the training becomes very slow. In our case, although the number of features is limited, the data is enough to make the classifier much slower than the others we used. On the other side it is considered a classifier with good performances able to distinguish well between classes as long as the features have a good discriminatory power. We chose to use SVC because, although its effectiveness is not always guaranteed, it has the potential to be the best.

Despite its name, the LR [12] is used for classification and not for regression. The intuition behind the LR is simple: it estimates the log probability of an event, based on available data. Although it also supports multi class problems, it is particularly suitable for binary classifications, as in our case. In theory, it is not necessary to standardise as with SVC, however we have achieved significantly better results by implementing the standardization step as well. It is a simple and effective classifier, though it is not among those with the best performance. It is also very sensitive to outliers and performs poorly with highly correlated features. As a classifier it is not among the most performing, but we have selected it for comparison with the other classifiers.

As representatives of neural networks we have implemented MLP [14]. The name is very explicit in explaining how it works. The perceptron is an algorithm that specializes in predicting by assigning to the features weights whose values are corrected every time the classification is wrong. In the MLP we add instead intermediate layers, called hidden layers. From the input we obtain a layer of values that are used to calculate another layer of values, always using weights. And so on until n layers are calculated. Neural networks are very popular classifiers, able to adapt to very complex and non-linear data types. It is worth testing one for our problem, although RFC and SVC should have an edge on MLP. On the negative side, setting the hyperparameters can be more complicated and is not among the fastest classifiers. As with SVC, it is necessary to use the StandardScaler function on the data. We wanted to use the MLP to test the effectiveness of neural networks.

One thing that distinguishes the classifiers is speed, in training and predicting. The training is the task requiring the most time. In the course of the project we have done it many times with all the classifiers. As will be shown later, the data set used has over 400 000 entries. Such a data size can lead to extremely long calculation times on low power computers. Luckily for us we had access to a server with 32 CPUs (Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz) from the University of Fribourg. Relying on such hardware, the time to train with 80% of the data set is about ten seconds for RFC, LR and MLP. Completely different is the case with SVC, where training with the 20% of the data set takes about 15 minutes to execute. The

largest size of the training set we tried out for SVC was 50% of the data set, and it took two hours before the end of the calculations.

3 The Data Set

In this chapter, we will describe the data sets generously provided by Armasuisse. The data set consists of two main files. The first is the data extracted from the Swiss Register of Commerce and includes information about companies such as name, address and more. The second contains the results from Google Search, complete with features used for machine learning. In order to explain the features we will dedicate a section of this chapter to them as well.

3.1 The Swiss Register Of Commerce

According to the definition given by the Federal Office of Justice [8] the Register of Commerce is the official source for economic information on Swiss companies. All Swiss companies are registered in it for legal reasons. Public information about companies can be accessed through the Central Business Name Index Zefix [18]. The information available from Zefix includes various kinds of data, such as the name of the company and its address.

We were provided with the data collected from Zefix. In total this data set contains information about 703 175 companies, for a total of 1.5 GB of disk space. The data consists of text files in which each row contains a JSON object related to a single company. The Table 3.1 shows a description and an example of the most relevant keys of the JSON objects in the database and it also indicates how often the keys have an assigned value. Most keys refer to string type data. The exception among the keys in the table is "address", whose elements are JSON type in themselves, whose keys represent city, street, number and other.

Key	**Frequency**	**Description**	**Example**
uid	703 175	Unique identifier for the company	CHE100135008
name	703 175	Company name	Auguste Reymond S.A.
purpose	697 576	Brief description of the business	La société a pour but...
address	703 175	Company complete address	{"street":"Zihlstrasse",...}

Table 3.1: The table shows the most important keys of the JSON objects containing company data are structure and what the frequency of each key is. There is also an example for each key from a company.

uid	domain	...
CHE100001141	books.google.dk	...
CHE100001141	books.google.dk	...
CHE100001141	books.google.dk	...
CHE100001141	www.moneyhouse.ch	...
CHE100001141	www.easymonitoring.ch	...
CHE100001141	www.cylex-swiss.ch	...
CHE100001141	www.tappeti-orientali.it	...
CHE100001141	biz1.ch	...
CHE100001141	ch.kompass.com	...

Table 3.2: All the domains found by Google while searching for the company with unique identifier CHE100001141.

Certain keys appear in each JSON object, indicating the information that a company must provide to the Register. On the contrary, other keys are rarer. By looking at the frequency of each key we can assess what are the most useful as they are available to as many companies as possible. For example, the address is available for all companies, so it could be a good idea to use it.

The Register of Commerce database serves as the basis for everything that comes next. In fact, on the basis of this data, searches are carried out on Google to find the site of each company and also to calculate the features used for the machine learning classification.

3.2 The Data Set With The Features Of Google Search Results

Armasuisse has in practice searched on Google for the names of a certain number of Swiss companies, saving about 10 first results for each company. Finally, for each of the results it calculated the features that will be explained later in Section 3.3. Armasuisse then provided us with the data set with the features corresponding to results so that we could use it for machine learning. Understanding how to read this data set was fundamental.

The data set is divided between several files in CSV format, it is therefore a table. The total size of the data set is 1.1 GB. Each row (which we will often refer to as an entry below) corresponds to a Google Search result. Since for each company there are between seven and eleven results, several entries refer to the same company, as shown in the example in Table 3.2.

The table has 28 columns. Among them, 22 refer to the features mentioned above. For example, if an entry refers to the fifth result by searching for the name of a company, the value for the column “position”, which is a feature indicating the position of the result, is 4. Or again, if the top level domain is “ch”, the value for the column corresponding to the feature “domain_suffix” will be the string “ch”. The remaining six columns are metadata and we list the three most important ones below. The “uid” column contains the unique identifier of the company. The “domain” column stores the domain returned as result from the Google search. The value in the “kompass_match” column is true if the domain is the correct one for the company, false if not and “?” if it is not known.

We want to determine for each entry whether the corresponding search result is actually the right website of the company. Then the columns “kompass_match” contains the ground truth, when available. Classifiers will be trained to learn how to predict the “kompass_match” column value based on the 22 features.

We conclude the section with some numerical data on the features data set, reported in Table 3.3. We highlight the most important information from the table. The data set contains 5 366 671 rows of which 438 814 are labeled and can be used for machine learning. Of the labeled entries, 61 166 (14%) are labeled

		Full data set	(%)	Labeled subset	(%)
Entries	Total	5 366 671	(100)	438 814	(100)
	Labeled true	-	-	61 166	(14)
	Entries with whois features	2 289 381	(43)	171 405	(39)
	Entries with website features	4 594 619	(86)	389 580	(89)
Companies	Total	562 986	(100)	48 274	(100)
	With all entries labeled as false	-	-	9120	(19)

Table 3.3: The features data set numbers.

as true, which means that the classes are unbalanced. Finally, for some companies the correct domain appears more than once while for 9 120 companies (19%) the correct domain is not between Google results at all (i.e. all rows have value "false" in the "kompass_match" column).

3.3 The Features

The task of the classifiers is to determine whether or not a Google Search result matches the website of the searched company. It is therefore essential to define features to distinguish each search result from the others, in the hope that they will allow to discriminate between the right and wrong sites. It is intended that the features actually have the ability to describe the site of a company.

Armasuisse already designed 22 features for us to use. For a complete description of all features refer to the Appendix A. They have been grouped into three categories. The first category is called "common features". It comprises the features that can be obtained directly by looking at the Google Search results, which contain a domain, a title and a description. Some features of this category are for example the position of the search result (i.e. first result: position 0 and so on), the suffix of the top level domain (i.e. ch or de), the similarity between the company name and the domain.

The second category is called "website features", and are the features obtained by analyzing the homepage of the domain indicated by the search result. In some cases the site is not accessible and therefore these features are missing. For example we have two boolean features that are true if the page contains the company name and respectively the town indicated in the company address.

The name of the last category is "whois features". It refers to features collected from an external site, Who.is [17], a database containing information about domains and IP addresses as the name of the owner. An example of features belonging to the category is the similarity between the domain owner's name on Who.is and the company name. As with the previous category, it is not always possible to get these features.

4 Solving The Companies' Websites Problem

In this chapter, we will define the problem and describe the method we use to tackle it. We will first define the problem formally, then explain the complete process, from obtaining the data to evaluating a Google Search result, including the operations that Armasuisse has carried out to provide us with all the necessary data for machine learning. Then we will focus on the work we have done ourselves, such as the optimization of the four classifiers described in Section 2.3, the features analysis and the integration of further improvements to the system.

4.1 Problem Definition

The aim of the project described in this report is to automatically identify if a result given by Google is in effect the website of a Swiss company using Machine Learning. A more formal definition of the problem is as follows. Let C be a set of Swiss companies. For each company $c \in C$ we have a set R_c which contains the first n_c results, obtained from Google Search by searching the name of company c. We aim to label each result $r \in R_c, \forall\, c \in C$ as $true$ when the result r matches the actual website of the company c, $false$ otherwise.

Armasuisse provided us with a data set of Swiss companies and a data set of Google Search results with associated features. We used the data obtained by Armasuisse to train classifiers and improve their performance with the help of various methods. As classifiers, we focused on the Random Forest Classifier, the Support Vector Classifier, the Logistic Regression and the Multilayer Perceptron. To measure performance we used precision, recall and F1 score. We compared the performance of these methods and selected the best ones.

4.2 The Workflow In Short

The ultimate goal is to obtain the websites of all Swiss companies. The process of achieving this goal is described below. Using the labeled entries from the data set with the results from Google Search and the features we train the classifier. Finally, the prediction is made using the supervised learning trained classifiers. Eventually, features are converted to numeric format if they are not already. Based on the

results of the classifier, a website is assigned to each company. The Algorithm 4.1 summarize the workflow described in this section.

Algorithm 4.1: How to find the websites of Swiss Companies

```
Result: Websites of Swiss companies
Train classifier with labeled results from Google Search with associated features;
foreach result from Google Search with associated features do
    predict if the result is the website of the company with the classifier;
    if the classifier predict true then
        assign the domain of the result to the company
    else
        the domain of the result is not the one of the company;
    end
end
```

4.3 Hyperparameters Tuning

Before training a classifier, it is possible to set parameters that affect the course of training. These parameters are called hyperparameters and are different for each classifier. Setting the hyperparameters correctly can positively affect performance and thus we will look for the optimal settings. Scikit-learn offers many hyperparameters to set for each of our four classifiers. However, only a small part of them has a significant effect. In the Table 4.1 we list the parameters we identified as worth to test, using the name they have in the Scikit-learn implementation of the classifiers.

Tuning the parameters means changing their values until we get the best results from the classifier. There are several typical techniques for conducting hyperparameter tuning. We've combined two of them. The first is the naive approach, manual tuning. It consists of manually modifying the parameters and then observing how the results of the classifier change. It is a simple, fast and intuitive method, and allows to easily visualize the trend of the classifier according to individual parameters. On the other hand, it requires to set everything manually and does not guarantee stability of the results, since each run of the classifier can be slightly different depending on random factors. The other method is called Grid Search and is the brute force approach. It consists of testing all possible combinations of the requested parameters. The function for Grid Search from Scikit-learn [13] also includes cross validation. Clearly this approach provides a reliable solution, however, it can be impractical due to very long calculation times for many parameters. We have therefore combined these two methods. At first, we tested individual hyper-parameters. Then we applied Grid Search to the most promising ranges and values from the first tests.

4.4 Features Analysis

In this work, under features analysis we basically enclose two types of analysis. The first is the analysis of individual features. In practice, we analyze the individual features to verify their quality and correlation with the others. The second is the analysis of the different combinations of features to be used to train the classifiers and their effects on performance.

Classifier	Parameter	Description
RFC	n_estimators	The number of trees in the forest.
	min_samples_leaf	The minimum number of samples required to be a leaf node.
	max_features	The maximum number of features considered when splitting a node.
	class_weight	Weight associated with the classes to predict.
SVC	kernel	The kernel used in the algorithm.
	gamma	The coefficient for certain kernels.
	C	The inverse of the regularization strength.
	class_weight	Weight associated with the classes to predict.
LR	solver	The algorithm to use in the classification problem.
	C	The inverse of the regularization strength.
	class_weight	Weight associated with the classes to predict.
MLP	solver	The solver for weight optimization.
	activation	Activation of the hidden layer.
	learning_rate	Learning rate schedule for weight updates.
	batch_size	The number of samples processed before updating the model.

Table 4.1: Important hyperparameters of the four classifiers.

4.4.1 Analysis Of Individual Features

We have conducted investigations on the 22 features using different approaches. First we looked at the features taken individually. Basically we trained each of the four classifiers with each single feature and compared the results. Then we classified them according to the F1 Score obtained with the testing set. In case of a tie, the feature is ranked first with better precision. We chose to classify them this way because a high F1 Score indicates a good balance between precision and recall and in general we give more importance to precision than recall. Thanks to the ranking then it is possible to see which features are best for the individual classifiers. Moreover, if all the classifiers agree on the best features we can say that they are the best features in general.

Another way we evaluated the quality of the features was to take the distribution of entries labeled true for each feature and compare it with the distribution of entries labeled false. For the comparison we used the independent samples t-test. In practice, if the two distributions are not significantly different, it means that the feature has no discriminating power. Moreover, the t-value obtained by t-test allows to classify the features. The higher the absolute value of the t-value, the greater the ability of the feature to discriminate between right and wrong website.

Finally, we also verified the correlation between all pairs of features, using Pearson correlation coefficient. In fact features with a strongly positive or negative correlation can be redundant and then it is possible to remove one of the two without losing in classifiers' performance. We also calculated the correlation of each feature with ground truth. A feature with high correlation to ground truth is considered useful.

4.4.2 Analysis Of Features Combinations

We tested different combinations of features determined by different techniques. First we tested the feature combinations proposed by Armasuisse. These are four different combinations based on the feature categories described in Section 3.3. In practice, it is a matter of excluding one or more categories of features, so to know what the performance of the classifier would be like if one of them was no longer available. The four combinations are "all set" (all categories of features), "common set" (only "common features"), "website set" ("common features" and "website features") and "whois set" ("common features"

and "whois features").

We then proceeded to use two automatic techniques to test the best combinations of features available in Scikit-learn. The first is Recursive Features Elimination [10]. This technique consists in ranking the features for the classifiers that support it, in our case RFC and LR, and recursively eliminating the worst ones. The process stops when the desired number of features is reached. We tested for all the features numbers between one and twenty-two and compared the results, eliminating only one feature each step. It is therefore a classifier-dependent method. On the contrary, the other method is independent of the classifier. This is a function called "SelectKBest" [11] and, as the name suggests, it selects a number of features considered the best. The function allows us to choose different alternatives to classify features. We have opted for the default one, the ANOVA F-Value between label and feature.

4.5 Other Optimizations

In order to push the results given by the classifiers further we have implemented and tested a couple of methods. For the first we consider the assumption that a company doesn't have more than one domain. For the other we collect ground truth from an external source.

4.5.1 No More Than One Domain

The first method to improve the results is based on the assumption that each company has a single website. As explained in Section 3.2, in the data set there are on average almost 10 entries for each company, corresponding to the first Google search results. Among them it may happen that the correct site appears zero, once or more times. There are therefore also many other results corresponding to wrong sites. However, when testing the classifier, one of the following errors may occur: for a company more than one site is predicted as correct; or the right site appears in multiple entries and is predicted once as correct and once as wrong. This type of errors are easily avoidable, so it is useful to develop a strategy to avoid them.

All the classifiers we have used allow to obtain for each entry the probability of assigning it a True label. If this probability is greater than 0.5, then the prediction will be True, otherwise False. The larger this value is, the more certain the classifier is in classifying the entry. We then implemented the following procedure. First we use the classifier to predict the entries as usual and get the probabilities for each of them. Then, for each company we take the entry with the highest probability ever. If it is still less than 0.5, for that company we keep the prediction of the classifier who has classified all entries as negative. If it is higher, we label all entries with the same website domain with the highest probability True. We also label all the other entries with the False label, even if they have a probability greater than 0.5. This way we don't assign more than one domain to each company.

There are considerations to be made about the choice of testing and training sets when testing this method. Selecting no more than one domain per company is more effective if there are many results to compare for each company. Usually during the split between training and testing sets entries are shuffled for better results. If there are 10 entries per company and a relative split of 80%-20% between training and testing sets, there will be an average of 8 entries in the training set, 2 in the testing set. This means that the method will not be as effective in the test phase. In the real case it will happen instead to have about ten entries for each company and the method could be more effective. For this reason we conducted tests without shuffling the features set and dividing it so that both the training set and the testing set contain about 10 entries per company.

4.5.2 Extract Ground Truth From Local.ch

The second method used to improve results is not related to classifiers or machine learning in general. We looked for another way to look for websites. There are many sites that contain information about

companies, details such as address, phone number, email address and website. One of the most famous in Switzerland is Local.ch [6]. Local.ch claims to contain information about more than 550 000 Swiss companies [7]. A company page on Local.ch contains several details, including the company's website when available. Using Local.ch has several advantages. First of all, our focus is on Swiss companies and with that site we are sure to find only Swiss companies. Moreover, it is possible to search companies by name and address, ensuring that when a result is found it is actually the correct one.

We have therefore developed a scraper that can search Local.ch for a company based on the data contained in the Register of Commerce. The scraper works as follows. First use the search function by entering the full company name and address. If there is a page on Local.ch with exactly that data then it is opened directly. This ensures that the page opened is actually the company's page and avoids false positives. Otherwise we get a list of possible results, but we consider that the company does not exist on the site. If the company page has been opened, then the scraper tries to extract the website indicated in the company data. If it succeeds, the search is completed. If not, the scraper checks if there is an email address in the form info@*domain*. If there is, then *domain* is considered as the company's website. Otherwise the search ends without result. The Algorithm 4.2 summarizes the behavior of the scraper.

Algorithm 4.2: Process carried out by the scraper looking for a company's website on Local.ch.

```
Result: A company website.
search company name and address in Local.ch search bar;
if obtain directly the company page then
    if company page contains company website URL then
        website found;
    else
        if company page contains company email then
            if email is the form info@domain then
                keep domain as result;
                website found;
```

Thanks to the scraper we have a second method to identify companies, in addition to machine learning: the ground truth collected from external sites. We have also implemented functions to measure the performance of a system that integrates both of these methods. In practice, we perform the prediction of the classifier. Then we modify the prediction for each company based on the ground truth collected on Local.ch, if available. Finally, we calculate precision and recall on the modified predictions.

5 Results

In the course of this chapter we will illustrate the numerical results obtained by implementing the tools, techniques and procedures described in Chapter 4. As a support for the information, multiple graphs and tables will be included. The data used for the plots is also available in Appendix C. We will start by describing the metrics used to compare the results. Then we will report the performance of the classifiers described in Section 2.3 without further input from us, so that we will be able to compare the progressive improvements. We will also vary the supervision degree to determine acceptable degree to train the classifier without losing in performance. Then there will be the results due to the various analyses and improvements, gradually stacked one on top of the other. It starts with hyperparameters tuning, continues with features analysis and ends with other types of improvements. In the last part we will discuss and summarize the most important results.

5.1 Metrics

The measurements we will report are precision, recall and F1 score. More precisely, precision refers to how many of Google's results classified as the correct site actually are. The recall instead refers to how many of Google's results with the correct site are classified as such. Accuracy is not taken into account as the data set is strongly unbalanced. Note that with each training run the scores may change slightly due to random factors such as the split between training and testing sets or the random nature of certain classifiers.

5.2 Base Results

To measure the effects of our work we need to consider what would be the performance of the methods to determine a company's website without our intervention. In concrete terms, we will illustrate the performance of Google search and classifiers without intervening on features and hyperparameters. We will also report the the results of testing the existing services.

A search on Google can return thousands of results, among which the desired site may appear. But we can only say that ideally the first result should be exactly what we were looking for. We therefore evaluate

Classifier	Precision	Recall	F1 Score
Google Search	0.7116	0.5586	0.6258
Base RFC	0.8478	0.9154	0.8803
Base SVC	0.8128	0.8876	0.8486
Base LR	0.7841	0.8390	0.8106
Base MLP	0.8246	0.8796	0.8512

Table 5.1: Performances of the base classifiers and of the Google Search

Service	Owner	Websites found
Company Name to Domain API	Clearbit	7
Domain Name Finder	Phantombuster	14
Company Data Enrichment	Powrbot	80
Company URL Lookup using Bing Search	Blockspring	84

Table 5.2: Websites found by existing services while searching for 100 companies.

Google's search based on how often the correct site is the first result of the search whose input is company name. In practice, the website is predicted as the right one if and only if it is the first result. According to this criterion, Google search provides 71.16% precision, 55.86% recall and 62.58% F1 score. Please note that these results also include companies of which the site is not known and perhaps does not exist. As base classifiers we mean Scikit-Learn's classifiers with default settings. No hyperparameter has been modified and all features have been used. For the RFC, LR and MLP we used 80% of the labeled features data set for training and 20% for testing. Those percentages are reversed for SVC, because it is a classifier that has difficulty with large data sets and training it with 80% of our data set entries takes too long. The data set was standardized and centered around 0 for the SVC, LR and MLP. The results are summarized in Table 5.1, including the performance of the Google Search. In fact, all the estimators perform better than Google Search. The RFC has an edge over the other classifiers. SVC and MLP are comparable, while LR is the worst one. This confirms the notion that the RFC is a good classifier for all occasions, directly out of the box. So this is the performance of the classifiers using them without implementing any improvement. In the following sections, we will illustrate how the adjustments made are reflected in the results.

But before proceeding, we report the results obtained by running the free version of the existing services mentioned in Section 2.2. For each of the four services we have searched for the URL of 100 companies whose website is known. Results are shown in Table 5.2 We believe that the observations made allow us to compare the various products and provide us with a benchmark to compare our results. In particular, the best results were given by programs that use a quality search engine or machine learning, namely the products from Powrbot and Blockspring. The service *Company URL Lookup using Bing Search* ranks first, having found 84 websites.

5.3 Varying The Supervision Degree

As stated in Section 3.2, the data set used for training and testing has 438 814 entries. A typical split between training and testing sets is 80%-20%. While most estimators can train with such a data set without problems, it is not necessarily the case that they all benefit by training with a large number of entries. For example, SVC is known to scale badly with large sets for memory reasons and its training can become very slow. This why in the last section we reported to have used only the 20% of the training set for that specific classifier. The analysis of hyperparameters and features requires training the classifiers many times. Being able to use a smaller training set speeds up this process considerably. We have therefore

		Testing Set Relative Size							
Classifier	**Score**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**
RFC	Precision	0.8532	0.8526	0.8498	0.8481	0.8478	0.8442	0.8407	0.8352
	Recall	0.9158	0.9154	0.911	0.9072	0.9016	0.8941	0.8896	0.8828
	F1 Score	0.8834	0.8829	0.8794	0.8767	0.8739	0.8684	0.8645	0.8583
SVC	Precision	-	-	-	0.8281	0.8152	0.8151	0.8136	0.8126
	Recall	-	-	-	0.841	0.895	0.8952	0.8968	0.8885
	F1 Score	-	-	-	0.8345	0.8532	0.8532	0.8532	0.8489
LR	Precision	0.8007	0.7972	0.7994	0.7989	0.7955	0.7997	0.7997	0.8035
	Recall	0.8467	0.853	0.85	0.8503	0.8511	0.8476	0.844	0.8378
	F1 Score	0.823	0.8241	0.8239	0.8238	0.8223	0.823	0.8213	0.8203
MLP	Precision	0.8171	0.8091	0.8203	0.83	0.8151	0.815	0.8156	0.8069
	Recall	0.8698	0.887	0.8577	0.8482	0.868	0.8617	0.8658	0.8702
	F1 Score	0.8427	0.8463	0.8386	0.839	0.8407	0.8377	0.84	0.8373

Table 5.3: Performance of the classifiers with different testing set size ratios.

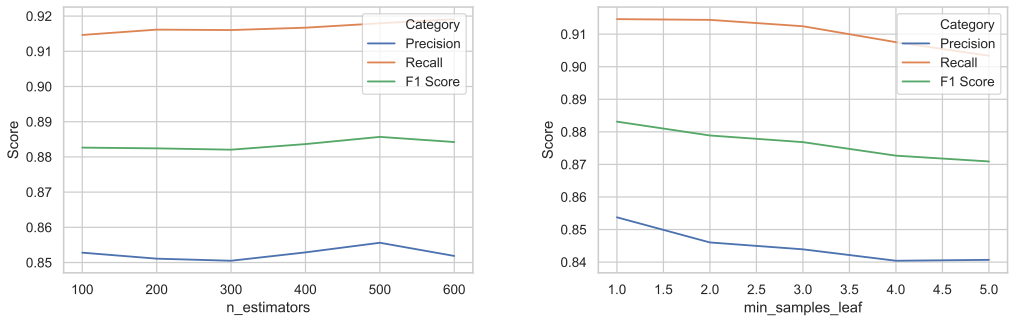
determined “acceptable” sizes for each classifier, i.e. sizes such that the performance remains roughly similar to that which would be obtained using a training set of relative size 80%. In addition, this analysis allows us to use larger testing sets and evaluate the consistency of the estimators even with external data sets.

In practice, we tested the performance of the basic estimators for a test set of relative size 20%, 30%, 40%, 50%, 60%, 70%, 80% and 90%. In the case of SVC we used only 50%, 60%, 70%, 80% and 90%, because using larger training sets is very slow. The results are shown in Table 5.3. On the basis of these results we can make a number of observations. The RFC is the only classifier with a clear tendency to worsen by reducing the size of the training set. This is because the RFC need the biggest number of examples to perform better. The other classifiers instead work fine with smaller training sets. The SVC is quite constant between 60% and 90% of the testing set size. It gets worse if the training set shrinks beyond that range, while it improves precision when enlarged. The LR shows great consistency even with small training sets. The MLP appears less regular, due to the fact that performance can vary widely. The best performances are obtained with a testing set relative size near to 50%, probably because it has the right balance to avoid overfitting and having a large enough training set. However, it seems that varying the size of the training set does not have a significant impact. In conclusion in the following it will be acceptable to split the training and testing set with respective ratio of 80%-20% for RFC and 20%-80% for all the others. When looking for the best performance as possible we will keep a respective ratio of 80%-20% for all classifiers but the SVC, for which we will use a ration of 50%-50%.

5.4 Hyperparameters Tuning Results

In this section we will show the results of the hyperaparameters analysis, one estimator after another. As explained in Section 4.3, we combined manual tuning and Grid Search. However, we did not use Grid Search for either SVC or MLP, as the calculations take too long, especially in SVC case. In general, we prioritized precision over recall.

For the RFC we analyzed the four following hyperparameters: “n_estimators”, “min_samples_leaf”, “max_features” and “class_weight”. Testing manually we have obtained that the default parameters are the best except for the number of estimators. In fact, it is worth increasing the number of trees around 500, even if the difference is very small. Increasing the number of trees gives better results, especially with larger data sets. It is negative to increase the number of minimum sample leaves because of the smoothing



(a) Number of estimators. Data: C.1

(b) Number of samples to be a leaf node. Data: C.2

Figure 5.1: How the performance of the RFC varies changing two hyperparameters.

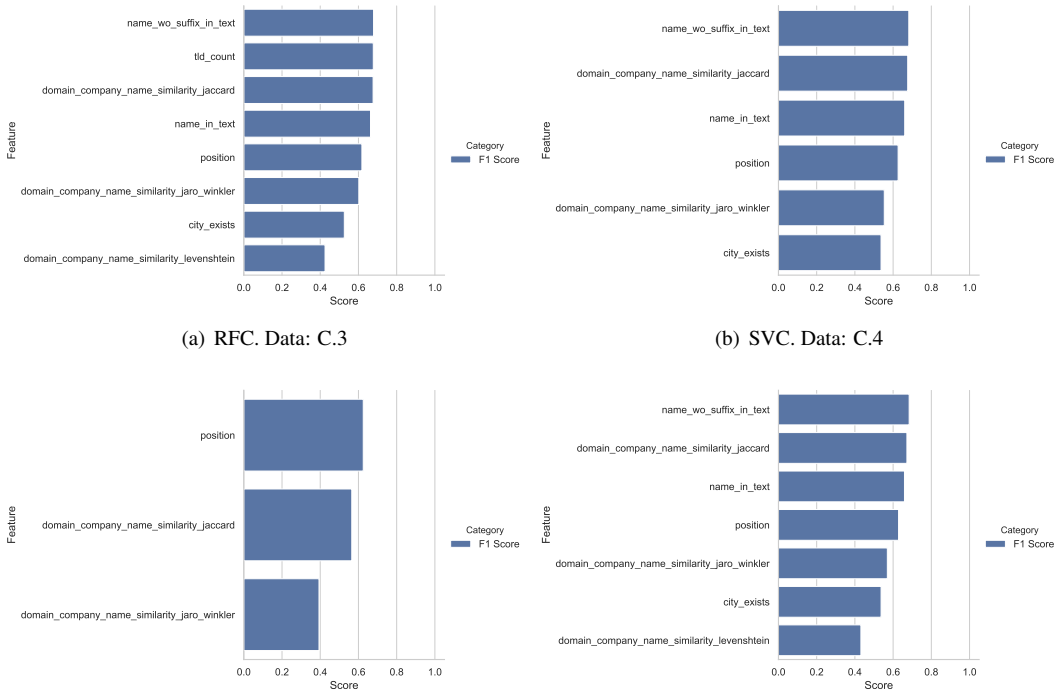
Classifier		Precision	Recall	F1 Score
RFC	base	0.8478	0.9154	0.8803
	improved	0.8526	0.9214	0.8857
SVC	base	0.8128	0.8876	0.8486
	improved	0.8149	0.8928	0.8521
LR	base	0.7841	0.8390	0.8106
	improved	0.7958	0.8345	0.8147
MLP	base	0.8246	0.8796	0.8512
	improved	-	-	-

Table 5.4: Comparison between classifiers before and after setting hyperparameters.

effect, the default is 1. To choose "max_features" it is better to use the default function, the square root on the total number of features. The difference is anyway minimal. Balancing the weights of the classes returns virtually the same precision but reduce the recall. As an example, the analysis results for the "n_estimators" and "min_samples_leaf" parameters are shown in Figure 5.1. Grid Search was therefore carried out with the most promising values. It confirmed that the default parameters are the best or at least equivalent to the others. We therefore concluded that the only parameter really worth changing is "n_estimators", setting the value to 500.

For the other classifiers the situation is similar. Most hyperparameters are best left at the default value, and even when this is not the case the differences are still small. The only parameter that is not definitely better with the default value for SVC is the regularization parameter. Still, the best value for it is between 1.0 (which is the default) and 10, with a slight preference for the latter. For the regularization parameter one has to test several orders of magnitude. In general, even with the worst performing values there is no huge difference. For the LR we selected the solver labeled "saga". It is a lot faster, but the performance is identical. The SAGA is almost identical to the SAG (Stochastic Average Gradient) solver with the difference that it supports L1 regularization. It is recommended for large data sets. Finally, for MLP it turned out that the default settings are actually the best.

We can therefore conclude that in our case, the default parameters in Scikit-learn do not need much change. Table 5.4 shows the comparison between the performance of the classifiers with the default parameters and those set by us. However, the differences are very small and can also be due to fluctuations in results every time the training is carried out.



(a) RFC. Data: C.3

(b) SVC. Data: C.4

(c) LR. Data: C.5

(d) MLP. Data: C.6

Figure 5.2: Features ranked by F1 score obtained by classifiers trained using only that features. Only the results with F1 score larger than 0.35 are reported.

5.5 Feature Analysis Results

As mentioned in Section 4.4, there are many results to be reported regarding the features analysis. For convenience, we will separate the following into two subsections. In the first we will report on our studies that include the analysis of individual features. It includes the study of the performance of the classifiers using a single feature, the comparison between distributions and the correlation between features. In the second one we'll report different combinations of features that we tested, including automated methods.

5.5.1 Ablation Analysis

With the various experiments we tested all the features. However, we will report here only the most relevant results, indicating the best and worst features.

The first one we report is the performance of trained classifiers with a single feature. We have therefore ranked the classifiers trained in this way according to the F1 score. That way we could identify the best features. In Figure 5.2 it is possible to observe the features ordered according to the F1 score obtained by the classifier trained only with it and that have obtained a score higher than 35%. The figure includes all classifiers.

Some observations can already be made from these results. The RFC scored more than 35% with 8 features, the SVC with 6 features, the LR with only 3 and the MLP with 7 But there seems to be consistency as to which features are better. According to these ranking, the best features are the position of the result from Google, if the name of the company or if the town of the company appear in the homepage

Feature	t-value (abs)
domain_company_name_similarity_jaccard	520.95
position	351.18
name_contained_in_path	292.33
domain_company_name_similarity_jaro_winkler	235.34
tld_count	212.00
name_wo_suffix_in_text	178.95
name_in_text	157.48
...	
whois_name_count	8.70
whois_organization_count	3.88

Table 5.5: Results of the independent samples t-test comparing the distribution of the features divided by ground truth label. All the distributions are significantly different. Only the most different and the most similar distributions are reported.

of the website and the similarity between the company name and the domain.Therefore we identify as important all the features of the "website features" group, i.e. the features that are collected directly from the homepage of the site. The others are the position of the site in the Google search result and the similarity of the company name with the domain. All the features of group "whois features" had generally lower performances instead, same for the features extracted from the description of Google Search results.

Another method we used to evaluate the features is comparing the difference between distribution, divided by label class. For all features the difference is significant according to t-test. We also used the t-value of the t-test to rank the features: the higher the absolute value, the greater the difference between distributions and therefore the feature is more important. The similarity between the company name and the domain using Jaccard metric is the feature with the most difference between distributions, with a t-value of 520.95. The feature counting how many times the owning organization according to Whois appears among all Google Search results is instead the one with the smallest t-value of 3.88. The features with the highest and smallest t-value are reported in Table 5.5. According to this ranking, the most important features are the same as those determined by testing each feature individually, with the addition of the number of times the domain appears among all Google Search results and if the name of the company appears in the Google result URL. In Figure 5.3 we can see the most distinct distributions ("position" and "domain_company_name_similarity_jaccard") and the least ones ("whois_organization_count" and "whois_name_count").

Let us also consider the correlation matrix, partially shown with most relevant features in Figure 5.4 as a heat map. It also includes the "kompass_match" label and therefore indicates the correlation between the features and the ground truth. The features with a stronger correlation, whether positive or negative, with "kompass_match" are once again the Jaccard similarity between the company name and the domain (0.62) and the position of the result (-0.47).In general, all the features identified above as important also have a relatively high absolute value of correlation with each other. Reverse situation for "whois features", in particular those of similarity between the name of the organization and that of the company. In fact, they have a correlation value close to zero, indicating a scarce usefulness. As for the pairwise correlation between features, there are no big surprises. The features belonging to the "website features" and "whois features" categories are strongly correlated within their group. This is also because not all entries have these features. Moreover, the similarities on the same subject are also strongly correlated. Outside of these cases the features do not display remarkable correlations.

From these studies we have learned that the features in the "website features" category are important, while those in the "whois features" category are less important. As for the "common features" category, the

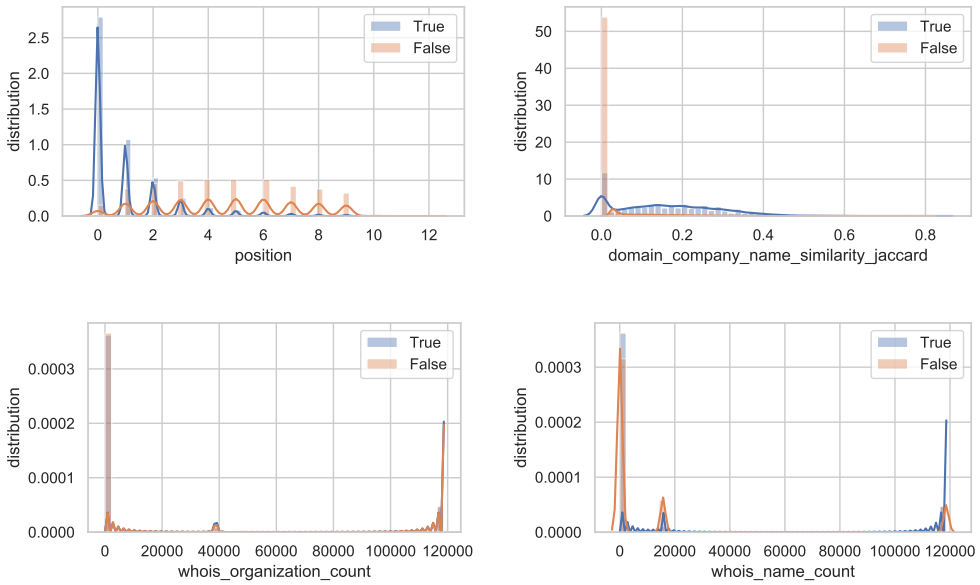


Figure 5.3: Distribution of features with respect to labels. On the top there are the two most different distributions, on the bottom the least ones.

most important features are the position of the result returned by Google, the number of time the domain appears among all the results from Google, if the name is contained in the path of the Google result and the similarities between the company name and the domain.

5.5.2 Results Of Analysis Of Features Combinations

Let's move on to the different combinations of features and their performance. Table 5.6 contains the results of the classifiers obtained using the subset of features proposed at the beginning of Section 4.4.2, namely the "common set", the "website set", the "whois set" and the "all set". In general, using all the features brings better overall results, while using only the commons ones is the worst option. The "whois set" and "website set" are a middle way. In general "website set" is a bit better in precision and "whois set" in recall. The exception is the MLP, which has a higher precision than the "all set" with the "website set", but this likely to be due to the fact that the classifier is highly unstable. Generally we can say that the RFC is the one that loses the most from changing the feature set. The other classifiers, while losing in performance, have a smaller loss. This confirms once again how the RFC is the classifier that most requires features and data, while the others are more stable.

The recursive feature elimination was conducted only on the two classifiers that support it, the RFC and the LR. It is in fact the classifiers that rank the features. We applied the method with the goal of finding the best k features for every k included between 1 and 22. The graphs shown in Figure 5.5 show how the classifiers perform using the features identified by the algorithm as the best. From the plot we can see how the two classifiers depend on the features in different ways. The RFC improves as the number of features used increases, with diminishing return effect. According to our test, the best case remains the one where all 22 features are used. The LR instead with 2 features is very close to the performance it would have with 22 features. This seems to confirm what we saw before, that only a few features are important for this classifier. From our calculations we have obtained that the best combination of features is a combination

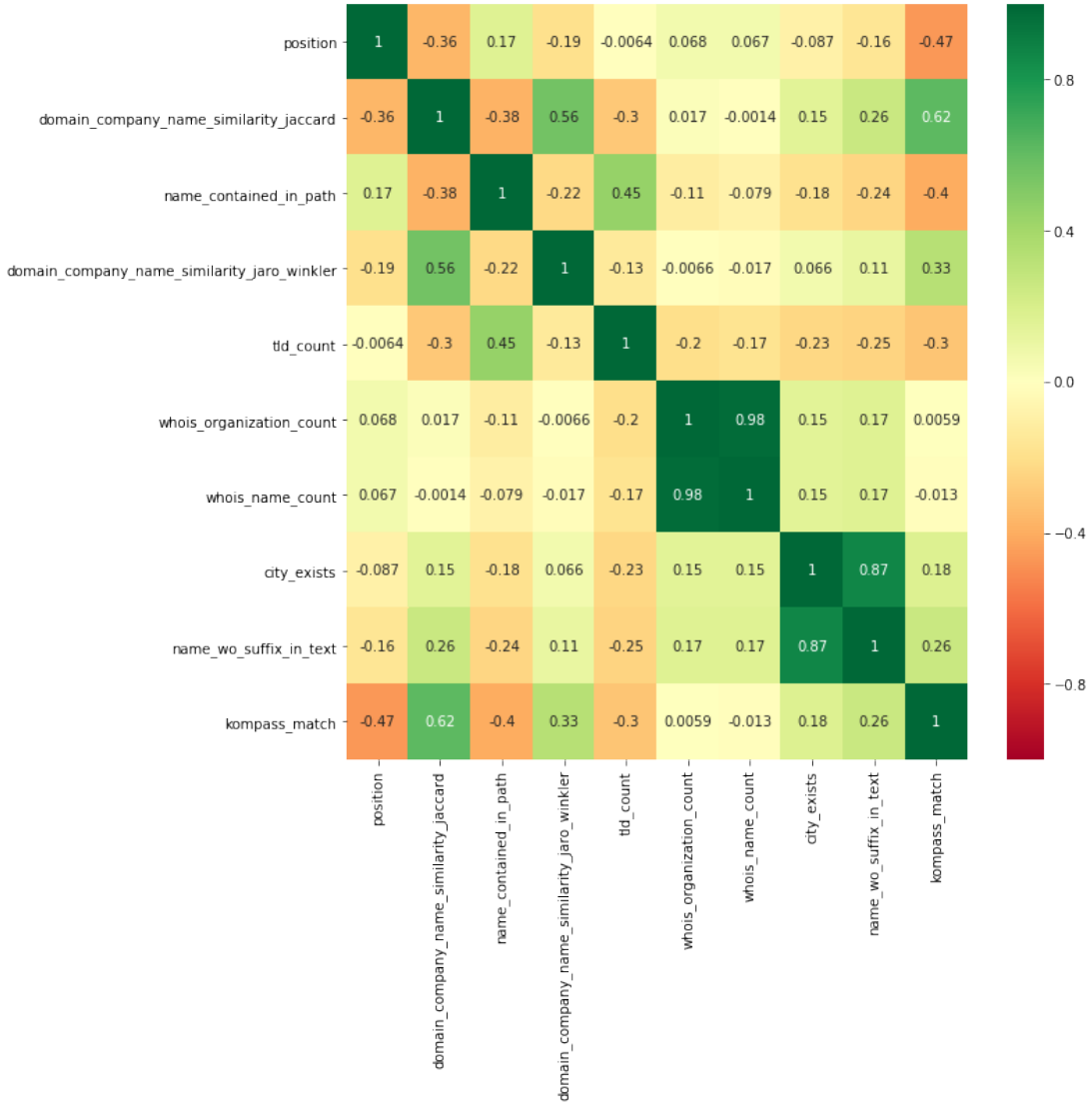


Figure 5.4: Correlations between features pairwise, including the label kompass_match. Only the most relevant features are included.

Classifier	Feature Set	Precision	Recall	F1 Score
RFC	all	0.8546	0.9205	0.8863
	website	0.8401	0.8978	0.8680
	whois	0.8353	0.9109	0.8715
	common	0.8251	0.8820	0.8526
SVC	all	0.8197	0.8853	0.8512
	website	0.8151	0.8783	0.8455
	whois	0.8042	0.8781	0.8395
	common	0.7964	0.8696	0.8314
LR	all	0.8089	0.8515	0.8296
	website	0.8006	0.8410	0.8203
	whois	0.8032	0.8506	0.8263
	common	0.7964	0.8411	0.8181
MLP	all	0.8107	0.9086	0.8569
	website	0.8248	0.8680	0.8458
	whois	0.7919	0.8945	0.8401
	common	0.8076	0.8605	0.8332

Table 5.6: Performances of the four classifiers using the subset of features labeled as ”all”, ”website”, ”whois” and ”common”.

of the 20 best features according to the algorithm. However, it is clear that the differences are negligible from the 8 features upwards.

The last test we did was to select the k best features according to SelectKBest function, using ANOVA f-value. As with the Recursive Features Elimination, we tested all possible numbers of features, but for all four classifiers. The graphs in Figure 5.6 show the performance of the classifiers with the different numbers of features. The trend is less regular than with Recursive Features Elimination as the selection of features does not depend directly on the classifier. According to the F1 Score, the number of k best features preferred is 20 for RFC, 22 for SVC, 10 for LR and 22 for MLP. However, it is interesting to note through the plots how, for all classifiers, the performance with the best k features with k between 10 and 14 are comparable, if not even better as in the case of the LR, to the performance with 22 features. However, further tests did not confirm the superiority of the LR results using only the 10 features indicated. The MLP's performance is as usual very erratic.

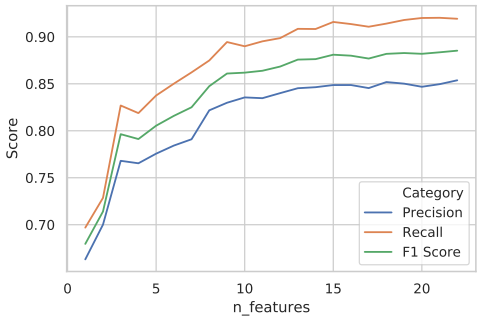
From these observations we saw how using all the features does not have a negative impact on the performance of the classifiers. The only classifier that works equally well with few features is the LR.

5.6 Other Optimizations Results

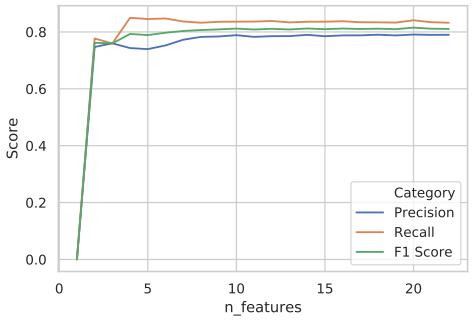
In this section we report the results obtained by implementing the techniques described in Section 4.5. This includes considering that each company as no more than one domain and looking for external sources of ground truth with the Local.ch scraper.

5.6.1 Results Of Assigning No More Than One Domain

The first method is to assign no more than a single domain to each company. Table 5.7 summarizes the results of this test. As can be seen, the table is divided into “Shuffle” and “No Shuffle” to distinguish cases where we have not shuffled training and testing sets, so we have about ten entries per company in the testing set as well. The right columns would therefore be more representative of a real case, where the

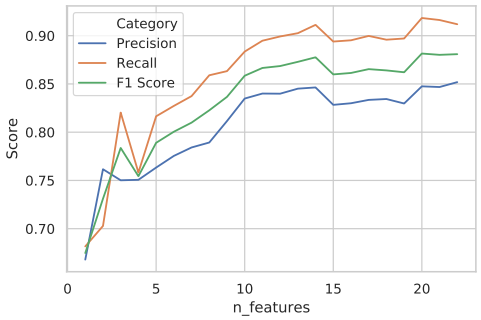


(a) RFC. Data: C.7

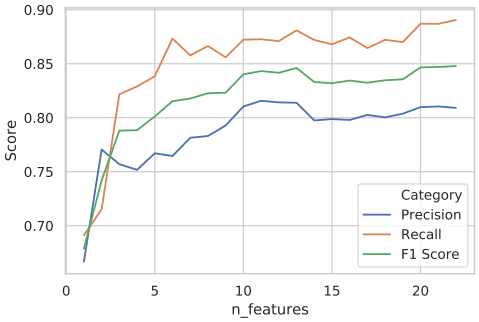


(b) LR. Data: C.8

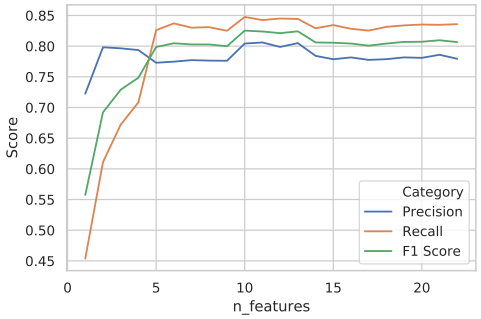
Figure 5.5: Performance of the classifiers with k best features according to Recursive Features Elimination algorithm.



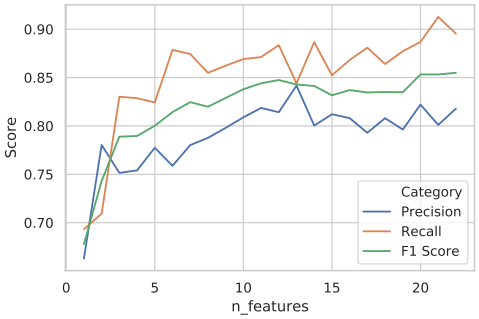
(a) RFC. Data: C.9



(b) SVC. Data: C.10



(c) LR. Data: C.11



(d) MLP. Data: C.12

Figure 5.6: Performance of the classifiers with k best features according to SelectKBest function.

		Shuffle			No Shuffle		
Classifier	**Improvement**	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
RFC	No	0.8546	0.9205	0.8863	0.8304	0.8868	0.8577
	Improved	0.8567	0.9211	0.8877	0.8608	0.9036	0.8817
SVC	No	0.8197	0.8853	0.8512	0.8152	0.8769	0.8449
	Improved	0.8543	0.8760	0.8650	0.8553	0.8748	0.8650
LR	No	0.8087	0.8515	0.8296	0.7778	0.8288	0.8025
	Improved	0.7991	0.8565	0.8268	0.8297	0.8801	0.8541
MLP	No	0.8107	0.9086	0.8569	0.8204	0.8726	0.8457
	Improved	0.8298	0.9047	0.8657	0.8523	0.9072	0.8789

Table 5.7: Improvement of the performance of the classifiers selecting no more than one domain for each company.

whole set of ground truth is used to train the classifiers and the set to predict contains about 10 entries per company.

The RFC and LR trained without shuffling data perform worse by default but gain a lot from the correction. The SVC and the MLP are a bit more stable even without shuffling the data set and they also get a huge improvement over the implementation of the correction. If we divide training and testing sets as usual the effect is very small for RFC and negligible for LR, which is even worse. The effect is greater for MLP, indicating that the predicted probabilities of the classifier are more accurate and high values are assigned to the right domains. For SVC the gain is very noticeable even with the usual shuffling split, but this happens because we used a different split between training and testing set (20% and 80%) and so the testing set contains about 8 entries per company.

Summing up, this type of correction works, proving that the predicted odds from the classifiers are reliable. In particular the SVC obtains a substantial improvement, bringing it to be comparable to the performance of the RFC that was in the lead so far.

5.6.2 Results Of Extracting Ground Truth From Local.ch

Since we were interested in measuring the effects of adding Local.ch among the sources, we did not search all the companies in the Register of Commerce. We've only searched for those contained in the data set of features whose domain is known. So we searched 39 150 companies on Local.ch. The scraper identified the website of 16 928 companies, which corresponds to 43.2% of the total. We also compared the domains found on Local.ch with those of Kompass, which act as ground truth. There are 927 domains that differ, which corresponds to 5.5% of the websites found. This difference is due to the fact that Kompass and Local.ch are not updated in the same way. Sometimes the different URLs lead to the same site, other times one of the two leads to an old company site. In general there is no rule that confirms which of the two sources is more correct. Anyway, by what we saw manually testing 10 pairs of URLs that differs between Local.ch and Kompass, we can assume that in most cases both addresses are correct. However, this will be negatively reflected in the following results, because certain companies will be assigned the Local.ch domain which differs from Kompass' domain. As a result there may be a loss in recall in our test, but the real value should be higher.

Adding the results of the classifier with the ground truth obtained by Local.ch gives improvements in precision and, as expected, small reductions in recall. The numerical results can be observed in Table 5.8. The improvement made by the results extracted from Local.ch does not depend directly on the classifier. Generally, the better the performance of the classifier, the less gain is made, because there are fewer entries to correct. In any case, the advantage is remarkable, considering that from Local.ch were obtained only

Classifier	Precision	Recall	F1 Score
RFC	0.8675	0.9105	0.8885
SVC	0.8409	0.8896	0.8646
LR	0.8247	0.8592	0.8416
MLP	0.8382	0.8940	0.8652

Table 5.8: Performance of the classifier when adding the information extracted from Local.ch.

	Shuffle			No Shuffle		
Classifier	**Precision**	**Recall**	**F1 Score**	**Precision**	**Recall**	**F1 Score**
RFC	0.8704	0.9091	0.8893	0.8642	0.8994	0.8814
SVC	0.8586	0.8920	0.8750	0.8660	0.8839	0.8748
LR	0.8221	0.8735	0.8471	0.8349	0.8892	0.8612
MLP	0.8413	0.9039	0.8715	0.8586	0.8950	0.8764

Table 5.9: Performance of the classifiers integrating the results from Local.ch and selecting no more than one domain per company altogether.

43.3% of the known websites. Finding multiple sources of ground truth to mix with classifier results looks a good way to push results even further.

Of course, we also combined the two methods. In this way, performance improves again. Performance is shown in Table 5.9. Again, we compared the results with Shuffle and without Shuffle. All classifiers perform better without shuffle except the RFC, which is due to the fact that it is less stable when changing the training set. Overall, the RFC is confirmed as the best, followed by the SVC.

5.7 Best Results

In this section, we report the very best results we obtained. We've done one last run of the classifiers with the best possible settings, so the numbers presented here may differ a bit from the ones from the previous sections. The Table 5.10 depicts these results and compares them to the base results reported in Section 5.2. We remind that the real recalls are likely to be higher because the 5.5% of results from Local.ch is different from the results from Kompass.

The RFC has proven to be a good classifier, consistently outperforming the others. We set the n_estimators hyperparameter to increase the number of trees in the forest from 100 to 500. The best results are obtained by using all the features and using as many training entries as possible. When adding the other performance optimizations of Section 5.6, it is better to Shuffle the features data set. The flaw of the RFC is that it doesn't seem to ensure its performance even in external data sets. In fact by reducing the training set or features the performance of the classifier is negatively affected. We cannot therefore ensure that the results presented here are comparable for real use case.

SVC is much slower than the other classifiers used when it comes to training. Due to technical limitations we were forced to use a relative ratio of 20%-80% to split training and testing sets. Although the relative size test of the training set in Section 5.3 suggested that increasing the size is useful, the last run conducted with a 50%-50% ratio returns results comparable to those in Section 5.6. This makes us deduce that the important thing for SVC is not to have so many entries for the training set, but to have so many entries per company. So the results of SVC, comparable to those of RFC, would be confirmed in a true use case, making it in the end the most reliable and promising classifier. As hyperparameters we set the regularization parameter to 10 and the kernel coefficient gamma to (1 divided by the number of features). It is also worth using all the features.

Classifier		Precision	Recall	F1 Score
RFC	Base	0.8478	0.9154	0.8803
	Best	0.8720	0.9124	0.8918
	Gain	+0.0242	-0.0030	+0.0115
SVC	Base	0.8128	0.8876	0.8486
	Best	0.8653	0.8931	0.8790
	Gain	+0.0525	+0.0055	+0.0304
LR	Base	0.7841	0.8390	0.8106
	Best	0.8351	0.8875	0.8605
	Gain	+0.0510	+0.0485	+0.0499
MLP	Base	0.8246	0.8796	0.8512
	Best	0.8599	0.8896	0.8745
	Gain	+0.0353	+0.0100	+0.0233

Table 5.10: Improvement in the classifiers performances.

The LR emerged as the weakest classifier. The performances are worse than the others. However, it does not lack advantages. The classifier is very stable whatever the training set is, and it also does very well without using all the features. For these reasons the effectiveness of the training set is guaranteed even if some features are missing. In any case, for the final run we have all the features and a relative ratio of 80%-20% between training and testing sets as they have no negative effects. The important hyperparameter to change is the solver. It was switched to "saga", which is a solver suggested for large data sets.

Finally, the MLP is the classifier in the middle. His performance is halfway between the others and like SVC and LR, the size of the training set doesn't influence the results much. However, results can fluctuate widely with each run of the classifier, making the results in a real case more uncertain. However, they should in any case be better than those of the LR. For MLP there was no need to change the default hyperparameters, and it also works well with all the features.

In conclusion, the RFC has the best performance, followed by SVC, MLP and finally LR. SVC is however the most promising as it is stable in performances with relatively small data sets, which is not true for RFC. This makes us deduce that the SVC performances are preserved also by applying the classifier to a real case with external data. The LR is the one that has benefited most from the proposed improvements, followed by SVC. The RFC had the smallest gain.

Finally, we can compare with the benchmarks reported in Section 5.2, i.e. the performance achieved by the existing services to automatically find the companies' websites. The most valid application that resulted from those tests was *Company URL Lookups using Bing Search by Blockspring*, which was able to identify the website of 84 companies while getting wrong 16. If we take our best classifier, based on recall we can say that on average it can assign the right website to the 91% of companies whose website is known to exist appears between the first Google Search results (for reference, this property was also true for the companies searched with the service offered by Blockspring). On the other hand, according to the precision, the 13% of all the sites assigned to the companies is wrong. We can therefore claim that our classifiers have even more promising results.

6
Discussion And Future Works

We will begin this chapter by discussing several idea that we tried to implement and then discarded following unpromising results. In the second part we will talk about future work that can be undertaken on the same subject.

6.1 Discussion

In the course of this work, we have experimented with different ideas. Until now, we have reported some of the most interesting and promising ones, which contribute effectively to solve the problem of ranking Google search results. But there are others that have proved to be less valid. This section is dedicated to failed ideas. The results we have achieved are far from perfection and can certainly be improved through further study. For this reason we want to support future work by pointing out directions in which it is not worth going in, so as not to waste valuable resources.

6.1.1 Uncertain Cases

The first idea discarded concerns the so-called uncertain cases. When predicting the class of a given set, classifiers assign each entry a number that indicates the probability of belonging to a given class. If the value is close to 0.5 then the classifier can be said to be insecure in determining the class of that specific entry. Uncertain cases are therefore those for which the probability is greater than $0.5 - \epsilon$ and smaller than $0.5 + \epsilon$, for ϵ of choice. Intuitively, most of the errors made by the classifier occur in these uncertain cases.

One technique that can be used in machine learning is to train the classifier with these uncertain cases so that he is better prepared to deal with them. We have therefore observed precision and recall of a subset of cases in certain cases. In our test we predicted 87 763 entries from the data set described in Section 3.2. We have obtained 1 903 uncertain entries by choosing $\epsilon = 0.1$.

The classifier was the RFC, so for the whole testing set the precision was around 85%, the recall 91%. If we consider only the 1 903 uncertain cases, precision and recall are actually much worse, around 58% and 62% respectively. We then took the uncertain cases and used 80% to train the classifier and 20% to test it. The results of the test are absolutely comparable to those obtained previously, with a 57% precision

and a 64% recall. In conclusion, therefore, the ability of the classifier to discern uncertain cases has not improved. We have therefore discarded this technique.

6.1.2 Scraping Websites For More Features

We also tried to collect more information directly from the websites found by Google. Most likely, this will be a way to improve the results of the classifiers, considering that some of the best features have been extracted directly from the websites. However, we were unable to obtain any useful information.

First we tried to get any email addresses contained in the homepage. Many emails from companies are in fact of the "info@companyname" form. The idea was therefore to find the similarity between the company name and the email addresses found, in order to get a new feature. However, we realized that this was not a good idea, since the website domain and the email coincide, thus developing a feature strongly related to an existing one. It could also lead to false positives, because sites like Local.ch often also contain the company's email address.

Finally, we even tried to access the impressum page of the companies' websites, with the intention of reading the information in there. It usually contains the name and address of the company, and sometimes even the UID. Finding the UID would directly confirm the correctness of the site without having to go through the classification. As a method to find the impressum page we simply added "/impressum" at the end of the domains. We tested 182 sites that we knew were correct. Among them we actually found the page in 67 cases (37%). In any case, looking personally at the found pages, we could not consider the information contained as useful. UID is very rare and is written in a different format each time, making it more difficult to find it. Other information such as name and address are generally also available on the homepage, making a potential feature redundant. So we've abandoned this option, too.

6.1.3 Adding Keywords To The Google Search

We then tried to intervene on Google search, to see if the use of different keywords helps to get the right site more easily. The hope is with better search results the classifier would get better performance more easily. Basically all the research was carried out using the company name as input. This could cause more famous companies with a similar name to appear first among the results.

As a criterion for evaluating the effectiveness of Google search, we considered whether the correct site is the first result or not. We moreover used only companies whose website is known. Applying these conditions, the search with only the names of the companies returns the correct site as the first result in 87% of cases, which is in fact a remarkable number. We have, nonetheless, tried to add some keywords to observe any improvements. However, we could not try large numbers of companies as our scraper was easily detected and blocked by Google.

As a first combination we looked for the name of the company combined with the name of the town. The scraper was able to do 724 searches. Among them, 78% returned the right site as the first result. In addition to the previous words we have also added the name of the street where the company resides. The number of correct sites has further deteriorated. Of the 542 searches made, only 67% returned the right site as the first result. As a last attempt we tried to add the word website to the company name. The result is particularly irrelevant, because the scraper was blocked after just 29 searches. Of these, 79% were correct.

In general, therefore, it seemed to us that searching only the name of the company is a good method. Adding more information increases the likelihood that sites such as Kompass and Local.ch will pop up among the first results.

6.1.4 Using Linkedin As Source Of Ground Truth

The last of the discarded ideas is related to expanding ways to get the ground truth, as we did by scanning Local.ch for links to company sites. The difference was to try another website, Linkedin. As well as offering a platform for workers, it is also possible to create pages for companies. Of course the page of a company on Linkedin can include a link to the official company's website.

We have therefore developed a scraper able to search the official page of a company on Linkedin first, and extract the address of the site then. To confirm that the company found on Linkedin is actually the correct one we only compared the name, if it is identical then it is the right one. Unfortunately, Linkedin imposes limitations on the search, which is about 1 000 queries per day. To have more at disposal it is necessary to subscribe to professional services. Nevertheless, we were able to search for about 900 companies on Linkedin. However, we were able to collect only 80 sites and a mysterious telephone number.

The result is worse than what obtained from Local.ch. Even if having many sources of ground truth is great, we would not channel the efforts in extracting data form Linkedin before other sources. Obviously there would be room for further improvements, like studying a technique to improve the certainty of finding the right page and not having false negatives. However, this would involve work of a certain complexity and in fact comparable to the work done to identify companies' websites.

6.2 Possible Directions

The automated search of Swiss companies' websites is a complex problem. As our results show, there is still plenty of room for improvement. However, we argue that this project can provide a solid starting point for future work. In this section we will briefly report the directions that seem most promising.

First, it can be useful trying to retrieve results from other search engines besides Google. As the result we obtained using *Company URL Lookups using Bing Search* (see Section 5.2) shows, Bing is also a reliable source. We could take the results from Google that also appear in Bing and add new features, such as the position of the result in Bing Search. Of course the same thing could be done with other search engines.

We also learned from the features study that the more features there are, the better, especially for RFC. So creating new quality features is very important. With this work we have provided a workflow to evaluate them. In particular, the analysis showed that the features extracted directly from the site are between the best ones. It may therefore be a good idea to focus on finding new features like this. An example that comes to mind is to use word embedding to see if the description of the company's activity in the Register of Commerce and the content of the site share the topics.

Finally, we strongly recommend to add new sources of ground truth like Local.ch and Kompass. We have seen that while getting relatively small percentages of ground truth from Local.ch the overall performance of the classification process improves. It is therefore worth investing in several sources. Even if the individual ground truth sources do not offer particularly outstanding results, combined with the others, it could be possible to arrive at fairly good percentages over the number of companies.

7 Conclusion

Our goal was to identify the website of Swiss companies using machine learning techniques. The contribution of this thesis consists of implementing and evaluating methods to solve the problem that have proved to be more effective than mere Google Search.With this report we documented the work done and the results obtained.

From a data set containing 22 different features provided to us by Armasuisse we trained four classifiers to predict whether a result obtained by Google searching for a company's name corresponds to its official website. The four classifiers chosen were the Random Forest Classifier, the Support Vector Classifier, the Logistic Regression and the Multilayer Perceptron. We then implemented several techniques to improve the performance of the classifiers. We experimented with different settings for the hyperparameters of the classifiers.We have carried out in-depth analysis of the features, discovering that those based on the information contained within the homepage of the domain under investigation are among the most important, along with the similarity between the URL and the company name, or the position among Google search results. We used automatic methods to find the best possible combinations of features. We have therefore implemented a kind of post processing to ensure that the predictions are consistent with each other, avoiding assigning more than one website to each company. Finally, we left the field of machine learning to venture into the world of web scraping, in order to increase our sources of ground truth. We then implemented a method to measure how the results of the classifiers sum up with those of the scraper.

In the end the classifiers that stood out the most are RFC and SVC, supported by the data collected by the scraper from `Local.ch`. The performance of these classifiers is such that 87% of the sites classified as company sites are actually correct, while 90% of company sites are correctly classified as such. For reference, if we classify sites as the company's if and only if they match Google's first result, these numbers are 71% and 56% respectively. Our classifier performed better than existing services, which obtained the correct website of the 84% of the companies from a small subset of 100 Swiss companies. In conclusion, our approach to automating the search of companies' websites has an edge over equivalent services.In particular, it outperforms other services in identifying small Swiss companies.

A
Features Description

The features used by the classifiers are described in detail below.

- Common features:
  - position (integer): the position of the result in the Google Search. The idea is that the right site is probably in the top positions.
  - domain_suffix (string): the top level domain, because a Swiss company will likely have ”ch” as top level domain.
  - tld_count (integer): the number of times the domain appears while googling the companies. Domains appearing many times will probably be databases about companies.
  - title_name_similarity_levenshtein (decimal): the string similarity distance between the Google result title and the company name based on Levenshtein metric.
  - title_name_similarity_jaro_winkler (decimal): same as the previous one but with Jaro Winkler metric.
  - title_name_similarity_jaccard (decimal): same as the previous one but with Jaccard metric.
  - name_in_description (boolean): true if the company name appears in the Google result description.
  - name_contained_in_path (boolean): true if the company name appears in the Google result url.
  - domain_company_name_similarity_levenshtein (decimal): the string similarity distance between the domain name and the company name based on Levenshtein metric.
  - domain_company_name_similarity_jaro_winkler (decimal): same as the previous one but with Jaro Winkler metric.
  - domain_company_name_similarity_jaccard (decimal): same as the previous one but with Jaccard metric.
- Website features:

  - city_exists (boolean): true if the name of the city declared in the Register of Commerce appears in the web page.
  - name_in_text (boolean): true if the name of the company appears in the web page.
  - name_wo_suffix_in_text (boolean): true if the name of the company appears in the web page without the suffix (e.g. SA, GmbH).

- Whois features:

  - whois_organization_count (integer): the number of time the organization of the domain appears between Google results.
  - whois_name_count (integer): the number of time the name of the whois owner of the domain appears between Google results.
  - whois_name_company_name_similarity_levenshtein (decimal): The string similarity distance between between the whois owner name and the company name based on Levenshtein metric.
  - whois_name_company_name_similarity_jaro_winkler (decimal): same as the previous one but with Jaro Winkler metric.
  - whois_name_company_name_similarity_jaccard (decimal): same as the previous one but with Jaccard metric.
  - whois_organization_company_name_similarity_levenshtein (decimal): The string similarity distance between between the whois company name and the company name based on Levenshtein metric.
  - whois_organization_company_name_similarity_jaro_winkler (decimal): same as the previous one but with Jaro Winkler metric.
  - whois_organization_company_name_similarity_jaccard (decimal): same as the previous one but with Jaccard metric.

B
The Project's Folder

Developing the program to run the classifier and improve them was the most important part of the project. The entire code is included in a folder as an attachment to this report. In this appendix we intend to explain its structure and use, and how to interpret it. The language used is Python, and most programs are in the form of Jupyter Notebook. The practical thing about notebooks is that they contain the most important results already calculated and displayed. If one wants to run the tests again, it is enough to run the wanted document. The project has been divided into 10 folders. We will explain below what the files in each folder are for. In general, the results of the calculations performed by the programs within a folder are in themselves contained in the same folder. Here are listed the folders by logical order:

- Data: This folder contains only the data sets, in particular the data from the Register of Commerce and the features set.
- Modules: This folder contains several functions' definitions used by the rest of the project. The most important ones are the functions to extract the desired data from the features data set, those to train and test a classifier and those to train and test a classifier including the optimizations described in Section 4.5. In addition, there are methods to display graphs and confusion matrices and some smaller utility to save results of tests in files.
- Data_Analysis: In here there are some analyses on the data sets contained in Data. The results collected in this folder are reported in Chapter 3 Also there is a document to extract the labeled data from the original features data set.
- Local_Scraper: This folder contains the only executable Python program that it is not a Jupyter Notebook. The program concerned is the Local.ch scraper. It takes as input a csv file with columns: company name, uid and address, i.e. the things used to do the search on Local.ch. Besides this in the folder there are a bash script to launch the scraper multiple times in parallel and two Jupyter Notebooks, one to get the csv files to use as input for the scraper and one to analyze the results of the scraper.
- Base_Performance: In this folder the efficiency of the classifiers has been tested without modifying hyperparameters or selecting features. The results collected in this folder are reported in Section 5.2

- Select_Test_Size: In this folder there are the tests done to measure the effects of changing training and testings size ratios. The results collected in this folder are reported in Section 5.3
- Hyperparameters_Tuning: This folder contains the tests performed by changing the hyperparameters to obtain optimal settings. The results collected in this folder are reported in Section 5.4
- Features_Analysis: This is the folder containing all the various experiments done with the features. The results collected in this folder are reported in Section 5.5
- Post_Processing: In this folder we measured the improvements made by assigning a single domain to each company and integrating the ground truth obtained from Local.ch. The results collected in this folder are reported in Section 5.6
- Best_Performance: This folder is parallel to the Base_Performance folder. In fact, it contains the best classifiers we could get. The results collected in this folder are reported in Section 5.7

C
Data Used for Plots

C.1 Changing n_estimators in RFC (Figure 5.1(a))

n_estimators	Precision	Recall	F1 Score
100	0.852801227935533	0.914643180508684	0.882640295484332
200	0.851099191659742	0.916158412620964	0.882431268113104
300	0.850504432895139	0.916035561409285	0.882054533925174
400	0.852883305584986	0.916707336912315	0.88364434687157
500	0.855614973262032	0.917957544463568	0.88569056185995
600	0.851871698690959	0.919137688940282	0.884227254668256

C.2 Changing min_sample_leaf in RFC (Figure 5.1(b))

min_sample_leaf	Precision	Recall	F1 Score
1	0.853755545357197	0.91461815798099	0.883139488883614
2	0.846031985515993	0.914390542193233	0.878884056267388
3	0.843920145190562	0.912435614422369	0.876841484973482
4	0.84041095890411	0.907544378698225	0.872688477951636
5	0.840687201998032	0.903383214053351	0.870908306872084

C.3 Single Features RFC (Figure 5.2(a))

We report only the results with F1 Score larger than 0.5.

Feature	Precision	Recall	F1 Score
name wo suffix in text	0.715047338909826	0.646579669187931	0.67909209951986
tld count	0.57497534659783	0.824762855716425	0.677581433503093
domain company name similarity jaccard	0.666430260047281	0.687058250060931	0.676587063482539
name in text	0.786116322701689	0.573371971343476	0.663098119530814
position	0.705009471690171	0.550180683311432	0.618045945197896
domain company name similarity jaro winkler	0.70488019344911	0.525139207337046	0.601877053026748
city exists	0.553712190650779	0.500870574579222	0.525967524269731
...	...	...	...

C.4 Single Features SVC (Figure 5.2(b))

We report only the results with F1 Score larger than 0.5.

Feature	Precision	Recall	F1 Score
name wo suffix in text	0.718860484090291	0.648982097186701	0.68213636705771
domain company name similarity jaccard	0.669303480594385	0.683693717652354	0.67642207312507
name in text	0.78353573227377	0.571236230110159	0.660751787441893
position	0.711836225709238	0.558875721541192	0.626149750339926
domain company name similarity jaro winkler	0.706714453360398	0.455189883761312	0.553727634194831
city exists	0.565538753644809	0.510852032586724	0.536806196228196
...	...	...	...

C.5 Single Features LR (Figure 5.2(c))

Feature	Precision	Recall	F1 Score
position	0.710029893825379	0.558230002431315	0.625045372050817
domain company name similarity jaccard	0.719084161529752	0.465396515225533	0.565073402204538
domain company name similarity jaro winkler	0.746954723052172	0.267094017094017	0.393486288516254
...	...	...	...

C.6 Single Features MLP (Figure 5.2(d))

We report only the results with F1 Score larger than 0.5.

Feature	Precision	Recall	F1 Score
name wo suffix in text	0.726078971533517	0.646419228253761	0.683937375659545
domain company name similarity jaccard	0.667394468704512	0.677141450607155	0.672232630121365
name in text	0.780058651026393	0.571145429019737	0.659451728247914
position	0.718440529552799	0.558735306039724	0.628602699744619
domain company name similarity jaro winkler	0.699274931653394	0.480755087031135	0.569782082324455
city exists	0.568800721370604	0.508504635227731	0.536965311768461
...	...	...	...

C.7 Recursive Features Elimination on RFC (Figure 5.5(a))

n_features	Precision	Recall	F1 Score
1	0.663053649407361	0.696852974922144	0.679533285383201
2	0.700086771318135	0.728533902479067	0.714027112916851
3	0.768027933809018	0.8269042170644	0.79637937819756
4	0.765388727661524	0.818790849673202	0.791189705534065
5	0.775625992889023	0.837526547949681	0.805388633596481
6	0.784244781925638	0.850133809099019	0.815861156510234
7	0.79095704596835	0.86216527024807	0.825027511397579
8	0.82175622542595	0.874835850295469	0.847465712581992
9	0.829821224800304	0.894464944649446	0.860931333859511
10	0.835519292359311	0.889944769330734	0.861873672618579
11	0.834688346883469	0.895204262877442	0.86388780677834
12	0.840058077334556	0.898634840186381	0.86835972984715
13	0.845300061236987	0.908515014397367	0.875768269955192
14	0.84638645297853	0.908323868245984	0.87626203334116
15	0.848668916385452	0.915837177308408	0.880974622450568
16	0.848685222205136	0.913590465154776	0.879942602040816
17	0.845438609862053	0.910755336617406	0.876882336666535
18	0.851862935807272	0.914097623635196	0.881883665091782
19	0.850166062801932	0.917929910350448	0.88274943177365
20	0.846836013828348	0.920062056013717	0.881931671428012
21	0.849607013301088	0.92026850032744	0.883527192706696
22	0.853742773481493	0.919240097008892	0.885281638055199

C.8 Recursive Features Elimination on LR (Figure 5.5(b))

n_features	Precision	Recall	F1 Score
1	0	0	0
2	0.747191011235955	0.777073732718894	0.76183945139169
3	0.759944230296072	0.759508196721311	0.759726150944943
4	0.743275062433107	0.849743045925443	0.792951206515947
5	0.73954891538572	0.845389605057886	0.788935289835638
6	0.752575812933869	0.847375349679118	0.797167073029142
7	0.772615223724509	0.83674459995146	0.803402205996582
8	0.782608695652174	0.832369942196532	0.80672268907563
9	0.783993311545185	0.835628645495787	0.808987882828124
10	0.78860662047729	0.836176638641743	0.811695257715622
11	0.782631981637337	0.836454329871617	0.808648563184315
12	0.785161090458488	0.838474898684972	0.810942686877575
13	0.785278442882521	0.833565630892842	0.808701877187401
14	0.789775367931836	0.835806213624068	0.812139073638934
15	0.784934828735981	0.835996771589992	0.809661533651216
16	0.78789959563592	0.837618622759348	0.811998741940556
17	0.787973692452239	0.834148363033568	0.810403832991102
18	0.790239726027397	0.833716538019379	0.811396148006074
19	0.787722007722008	0.8328706727629	0.809667433923327
20	0.7908178777744	0.840999110823701	0.815136913855917
21	0.789623805118717	0.834256393549438	0.811326732673267
22	0.789979835582441	0.832325543389443	0.810600031831928

C.9 SelectKBest on RFC (Figure 5.6(a))

n_features	Precision	Recall	F1 Score
1	0.667927869908227	0.68159040499466	0.674689977637731
2	0.761612931876721	0.70281124497992	0.731031543052003
3	0.750149298297999	0.820326530612245	0.783669968026203
4	0.750644745325596	0.758160358160358	0.754383833475074
5	0.763313609467456	0.816455696202532	0.788990825688073
6	0.775409329626197	0.827222542638214	0.800478373529998
7	0.784162792470086	0.83738911044193	0.809902392947103
8	0.789299420573407	0.858979608549668	0.822666666666667
9	0.811529933481153	0.863277755185035	0.836604398202885
10	0.834865960099751	0.883546391752577	0.858516648635653
11	0.840043106766223	0.894801574286651	0.86655814507484
12	0.83990352001206	0.899209167204648	0.868545149849955
13	0.845113239094087	0.902597402597403	0.872909961535442
14	0.846373329328728	0.911236863379143	0.877608221737776
15	0.828334348355664	0.893936904370687	0.859886201991465
16	0.829988810145468	0.895236562600579	0.861378856501374
17	0.8334722959145	0.899762703543082	0.865349807192886
18	0.834376649822762	0.895862013118471	0.864026866604186
19	0.829726264987558	0.897105584997962	0.86210138682128
20	0.847513389441469	0.918338583982756	0.881505650167118
21	0.846806883365201	0.916252896391923	0.880162168607655
22	0.851874521805662	0.911875511875512	0.880854430379747

C.10 SelectKBest on SVC (Figure 5.6(b))

n_features	Precision	Recall	F1 Score
1	0.666738677874229	0.691115996905663	0.678708516593363
2	0.770539867292727	0.715361630715075	0.741926240382064
3	0.75697016412551	0.821781369907747	0.788045446093926
4	0.751733086190917	0.828993091042884	0.788474998055836
5	0.767079699022948	0.83842062193126	0.801165132054816
6	0.764535196131112	0.873329923273657	0.815319230218232
7	0.781447946940905	0.857718340388115	0.817808713284395
8	0.783048598440869	0.866450507769485	0.822641071065548
9	0.792833950011327	0.855931685246703	0.823175452523055
10	0.810395467379152	0.872279671214995	0.840199603151215
11	0.815705067012868	0.872686792761142	0.843234388568722
12	0.814205250596659	0.870943365398362	0.841619135960844
13	0.813891564217325	0.88095335431912	0.846095705521472
14	0.797493416014494	0.872061436653663	0.833112195121951
15	0.798795180722892	0.867978562369595	0.831951061702253
16	0.79793926044836	0.874388921843359	0.834416661786809
17	0.802630332202619	0.864561432158239	0.832445606242261
18	0.800333370790725	0.872155438088046	0.83470226875409
19	0.803753816002713	0.870058341152952	0.835592821879163
20	0.80976672313965	0.887073747756567	0.846659204827486
21	0.810459718478586	0.886981460616614	0.846995764825556
22	0.809126850919399	0.890519488516246	0.847874305033075

C.11 SelectKBest on LR (Figure 5.6(c))

n_features	Precision	Recall	F1 Score
1	0.722418070244157	0.45382217847769	0.557453025036522
2	0.798159246575342	0.610960924059966	0.692125655422022
3	0.796432725862737	0.672065439672802	0.728982742558005
4	0.793614490897448	0.708394577821329	0.748586961211546
5	0.7729494839761	0.826022058213782	0.798604986771426
6	0.774721133554417	0.837052117263844	0.804681384061375
7	0.777223984292403	0.830133892563317	0.802808112324493
8	0.776418412601315	0.830864904672285	0.802719475078066
9	0.776171063764326	0.824967298888162	0.799825625173384
10	0.804195264903216	0.847598146793465	0.825326474079937
11	0.806011143372832	0.842506767287343	0.823854977139649
12	0.798698315467075	0.844957472660996	0.821176933674474
13	0.804963708733318	0.844302554027505	0.824163969795038
14	0.78418096483804	0.829089725860246	0.806010280743377
15	0.7786395718607	0.834358377659574	0.805536609829488
16	0.781636573184787	0.82828529555447	0.804285093094043
17	0.777597402597403	0.825319986872333	0.800748288489094
18	0.778761737537217	0.831445105550575	0.804241564175339
19	0.78166909341712	0.833810185563639	0.806898188434459
20	0.780935555726246	0.835209069251622	0.807161003493172
21	0.785913744958114	0.834733893557423	0.809588493807431
22	0.779373368146214	0.835790167174504	0.806596463341943

C.12 SelectKBest on MLP (Figure 5.6(d))

n_features	Precision	Recall	F1 Score
1	0.662919594067135	0.693224489795918	0.677733439744613
2	0.780300982247454	0.709288990825688	0.743102338553958
3	0.751502114086492	0.830205687126117	0.788895810621399
4	0.754054851076379	0.828647816576197	0.789593546145829
5	0.777444418801446	0.824306688417618	0.800190031275981
6	0.758764785041434	0.87854682630802	0.814274313077186
7	0.780156926010119	0.874342537804076	0.824568881999612
8	0.787667640668315	0.854854447877704	0.819886917527783
9	0.797969775606778	0.862268041237113	0.82887382566298
10	0.808872430471584	0.869102720259846	0.83790660351509
11	0.818589792348064	0.871134855107657	0.844045331555625
12	0.814152644230769	0.883508600309774	0.847413894210094
13	0.84163628999595	0.844029244516653	0.842831068748733
14	0.800382887858037	0.886623164763458	0.84129871135018
15	0.812050984936269	0.852278255229447	0.831678468293841
16	0.808056333762399	0.868136337753193	0.837019607843137
17	0.792905081495686	0.880868496517821	0.834575376494333
18	0.808036739380023	0.863982322612325	0.835073564309445
19	0.796225852166778	0.87722902457063	0.834766969033469
20	0.822084498526188	0.88683245006115	0.853231879510511
21	0.801032850380146	0.912641987415216	0.85320294892853
22	0.817976852547723	0.895360315893386	0.85492105883277

Acronyms

LR Logistic Regression.

MLP Multilayer Perceptron.

RFC Random Forest Classifier.

SVC Support Vector Classifier.

References

[1] 3.2.4.3.1. sklearn.ensemble.RandomForestClassifier. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html, Last accessed on 2020-08-02.

[2] Company Name to Domain API. https://clearbit.com/blog/company-name-to-domain-api/, Last accessed on 2020-08-09.

[3] Company URL Lookups using Bing Search - Blockspring. https://open.blockspring.com/lists/browse/enr01CJQ1MD3FCZC0HXN9M3W8CQZC, Last accessed on 2020-08-09.

[4] Domain Name Finder — Phantombuster. https://phantombuster.com/automations/toolbox/3171/domain-name-finder, Last accessed on 2020-08-09.

[5] Find company information with our database search. https://powrbot.com/company-search/, Last accessed on 2020-08-09.

[6] local.ch - the official phonebook and yellow pages of Switzerland. https://www.local.ch, Last accessed on 2020-07-31.

[7] local.ch - Your platform for bookings and information. https://info.local.ch/en, Last accessed on 2020-08-24.

[8] Registre du commerce, Zefix® et Regix. https://www.bj.admin.ch/bj/fr/home/wirtschaft/handelsregister.html, Last accessed on 2020-06-15.

[9] scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/index.html, Last accessed on 2020-08-01.

[10] sklearn.feature_selection.RFE. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html#sklearn.feature_selection.RFE, Last accessed on 2020-07-29.

[11] sklearn.feature_selection.SelectKBest. https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectKBest.html#sklearn.feature_selection.SelectKBest, Last accessed on 2020-07-29.

[12] sklearn.linear_model.LogisticRegression. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.

LogisticRegression.html#sklearn.linear_model.LogisticRegression, Last accessed on 2020-08-02.

[13] sklearn.model_selection.GridSearchCV. https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html, Last accessed on 2020-07-27.

[14] sklearn.neural_network.MLPClassifie. https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, Last accessed on 2020-08-02.

[15] sklearn.preprocessing.StandardScaler. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, Last accessed on 2020-08-02.

[16] sklearn.svm.SVC. https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html?highlight=svm#sklearn-svm-svc, Last accessed on 2020-08-02.

[17] WHOIS Search, Domain Name, Website, and IP Tools - Who.is. https://who.is/, Last accessed on 2020-08-10.

[18] Zefix – Central Business Name Index. https://www.zefix.admin.ch, Last accessed on 2020-06-15.

[19] Silviu Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, pages 708–716, 2007.

[20] Mark Dredze, Paul McNamee, Delip Rao, Adam Gerber, and Tim Finin. Entity disambiguation for knowledge base population. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 277–285, 2010.

[21] John B Killoran. How to use search engine optimization techniques to increase website visibility. *IEEE Transactions on professional communication*, 56(1):50–66, 2013.

[22] Yubin Kim, Kevyn Collins-Thompson, and Jaime Teevan. Crowdsourcing for robustness in web search. In Ellen M. Voorhees, editor, *Proceedings of The Twenty-Second Text REtrieval Conference, TREC 2013, Gaithersburg, Maryland, USA, November 19-22, 2013*, volume 500-302 of *NIST Special Publication*. National Institute of Standards and Technology (NIST), 2013.

[23] Cheng-Jye Luh, Sheng-An Yang, and Ting-Li Dean Huang. Estimating google's search engine ranking function from a search engine optimization perspective. *Online Information Review*, 2016.

[24] David Milne and Ian H Witten. Learning to link with wikipedia. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 509–518, 2008.

[25] Mohammad Moradi. Crowdsourcing for search engines: perspectives and challenges. *International Journal of Crowd Science*, 2019.

[26] Albert Weichselbraun, Daniel Streiff, and Arno Scharl. Consolidating heterogeneous enterprise data for named entity linking and web intelligence. *International Journal on Artificial Intelligence Tools*, 24(2):1540008, 2015.