UNIVERSITÀ DEGLI STUDI DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in Informatica

...

Elaborato finale

Machine Learning Classifier: Matching Italian Companies with Facebook Pages and Users

A Random Forest Approach

Yannis Velegrakis

.....

Michele Sordo

.....

Anno accademico 2015/2016

Acknowledgements

...thanks to...

Indice

Summary

Nowadays companies are getting more and more connected with potential customers with the help of social networks. These instruments are not only supposed to help people getting linked and chat together but also to give public figures and companies a way to show themselves and to get promoted. Social Networks directly provide services for advertisement: Facebook Ads for Facebook, Twitter Ads for Twitter etc. These services are developed to help people and organizations promoting themselves on that particular platform and usually aren't very useful as a platforms itselves to manage an entire lead generation campaign. Companies are supported only in the brand and products promotion, meaning that the ones that want to find potential customers are generally left alone. The lead generation process, the generation of consumer interest or inquiry into products or services of a business , requires company to have a list of potential customers and, most important, a way to connect with them.

The aim of this project is to show a practical solution to connect real companies with their social pages. Such solution is meant to be implemented in a bigger platform for lead generation that will help companies in the path of finding other businesses that may be interested in buying specific products. The solution shown connects Italian companies with their Facebook page in order to enrich companies' description on Atoka , a leading tool developed by SpazioDati that collects structured data and information of all the 6 million companies currently opened in Italy.

0.1 Metodologie e Tecniche Usate

0.2 Dati elaborati

0.3 Conclusions

1 Introduction

1.1 Introduction

One important part of companies' public relations is being connected with potential customers and make it easy to be found in the huge quantity of data available by potential clients. In a connected world, where many companies have a Facebook page (and many a website too), it's difficult to find somebody interested in buying your product or solution. As introduced before, social networks provide advertising services that are vertical to the particular platform (ex. Facebook, Twitter, etc.) but a marketing platform for lead generation service needs to link social networks' data with existing companies legally registered in the real world.

The paper analyze the process of connecting Italian companies with their Facebook pages or user profile. The aim of the project is to create a model that can automatically find if a Facebook page is the official (and real) page of a determined company. This job is very simple if it's done by a human but it can't be done by people because of time lacking: nobody could find the right match for more than 6 million companies in a reasonable amount of time. The only way to overcome human computational time and cost is to develop a prediction algorithm that can learn by itself to recognize an official page (or user) from a set of potential pages (and users).

The paper analyze only the subset of companies that has ATECO code n.56. ATECO n.56 is the code that collect companies that operate in the food sector . This limitation is caused by time reason: the classification time that the VPS will spend analyzing all the 6 million companies will probably be more than 20 days. Apart from time causes the process needs to be integrated with Spaziodati's

information retrieval pipeline and the automation process made with Azkaban Flow is not described in this paper.

This problem was an important step in upgrading company information available on Atoka and gained priority at SpazioDati, making the company invest money and time to develop this project. This approach might also be extended to different information and usages in similar problems. The solution developed was born after discussion with SpazioDati's developers, that have big experience in comparable problems (like websites classification etc), taking care of the previous problem solution and hurdles found developing that.

The final solution is a machine learning algorithm application that runs after a score generating application that compute how similar is a page/user data collected to the company data already suited in Atoka Index. The machine learning algorithm chosen to classify the data in this project is Random Forest Classifier and the solution will explain why, in this situation, a random forest approach is better than other classification algorithms.

2 Motivation

2.1 Motivation

Before the project, only 1.62% of Italian companies had a Facebook Page/User associated in Atoka Index. The previous knowledge about facebook pages/user was collected analyzing the content of companies' official websites but was clearly not enough for a business product that aims to connect companies to new business customers.

The subset chosen is interesting to study because many restaurant, pubs and companies involved in the food sector:

- have a Facebook Page, or user created erroneously instead of a business page (now Facebook has built a tool that migrates a business user account to a formal Facebook page but not all the companies have used it so far).

- still don't have a website: this is a really fundamental task to improve contacts in Atoka and a great reason to find how many of them really have a Facebook Page.

Another interesting part of the problem is how to sample a training dataset for the classifier. In fact, as said before, there are only few companies with ATECO n.56 connected to a website and this implies that less of them have a Facebook page/user connected. The number of matched Facebook pages is 6,355, over a dataset of 362,154 companies with ATECO n.56, less than 2% of them.

The result that this project will produce will be used to upgrade Atoka index data and, in consequence, data available on the online Atoka lead generation platform.

3 Problem Statement

3.1 Problem Statement

Atoka Index stores information about Italian companies, including contacts like emails, official website and official Facebook page. Only the 1.62% of the companies had a linked Facebook Page or user before the project. The aim is to fill more companies with Facebook social data. This topic was chosen because in today's world it has become more and more important to have social information about companies if you want to find potential customers through a lead generation technique.

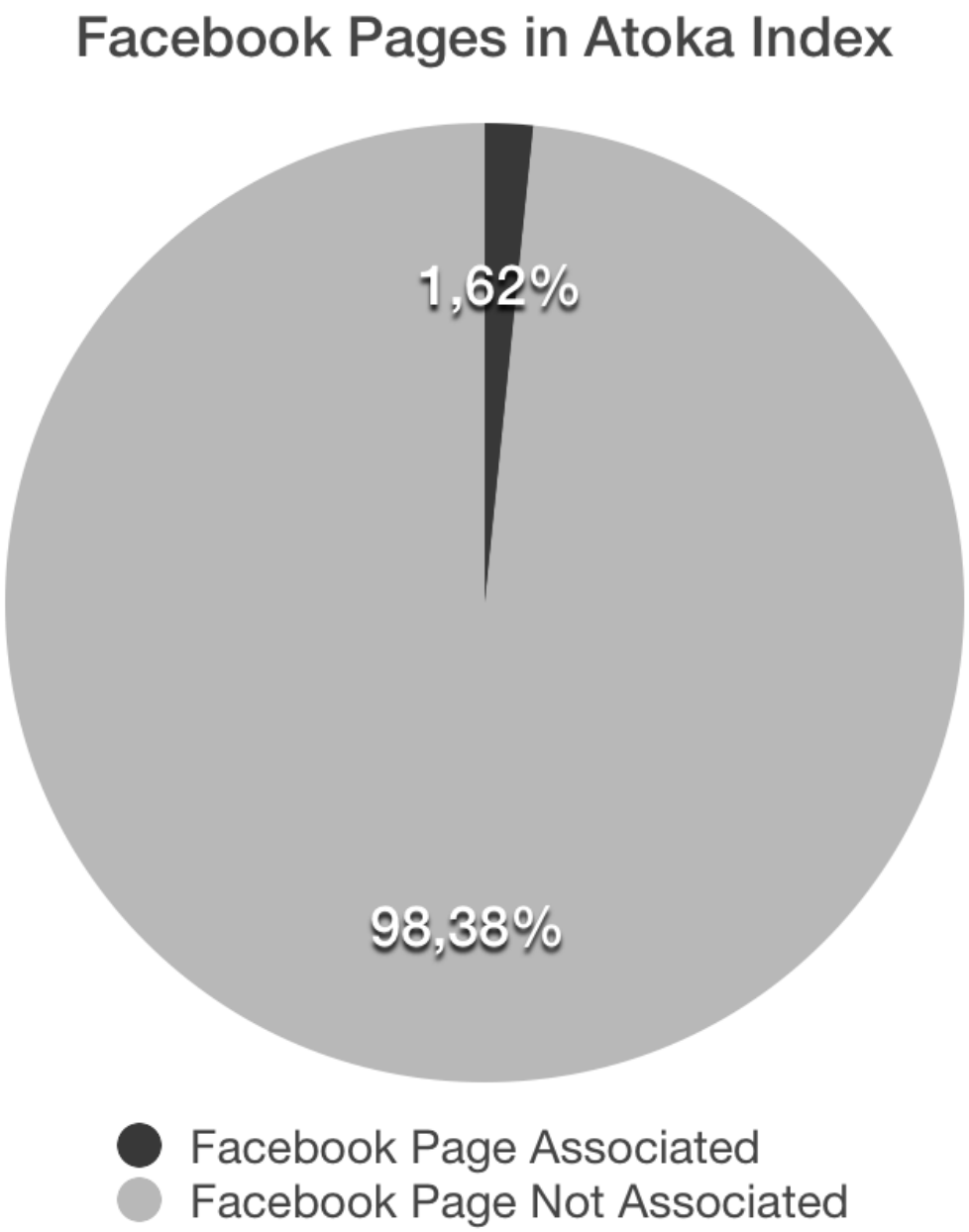

Figura 2.1: Only 96310 over 5952128 companies in Atoka had a Facebook Page or User associated before this project.

3.1.1 Data Sources

The data sources used in the project are mainly two: Atoka Index and Facebook's GraphAPI. Data is analyzed and collected in a single index, used by the classifier to predict if a Facebook page belongs or not to a particular company.

Atoka

Atoka is a collection of data coming from Cerved Group S.P.A and crawlers made at SpazioDati, stored in a Elasticsearch index. This elasticsearch index is the heart of atoka.io, the lead generation tool in which the results of this project will be shown, added to data previously available.

The most important fields of Atoka index used in the project are:

- ateco: ateco code of the company, representing the classification of economical activities
- description: a description of the company, used as a source for the research of keywords
- emails: list of email of the company
- entities: entities extracted with Dandelion API
- location: list of addresses of the company (companies may have more than a single office)
- social: social media link of the company (facebook, twitter, linkedin, etc.)
- website: official website of the company

It's important to underline the fact that not all the companies have a social and a website stored in Atoka index: the purpose, in fact, is to fill the missing values in social field.

GraphAPI

The Graph API is the primary way to get data in and out of Facebook's social graph . It's a low-level HTTP-based API that you can use to query data, post new stories, upload photos and a variety of other tasks that an app might need to do. The endpoint used for retrieving information is:

```
GET graph.facebook.com
  /search?
    q={your-query}&
    [type={object-type}](#searchtypes)
```

After being cleaned, the company name is put in the parameter q as q={your-query}, for each company pages and user are searched inside Facebook graph (because many companies still use user instead of pages). For making less number of GraphAPI calls and reducing the HTTP request time we used a batch request, collecting multiple calls in only one. The limit imposed by Facebook is a maximum of 50 call per batch request. In this project, every batch request was made with 50 request.

The Batch endpoint is:

```
curl \
    -F 'access_token={TOKEN}' \
    -F 'batch=[{ "method":"POST", \
                 "relative_url":"search", \
                 "body": {query_1} }, \
               { "method":"POST", \
                 "relative_url":"search", \
                 "body": {query_2}]' \
    https://graph.facebook.com
```

3.1.2 Problem Definition

Achieving the required functionality seems to be a three-step process.

First is to take companies' names from Atoka index, clean them and create a new elasticsearch index with Facebook responses. Facebook responses should contain the same information types of Atoka in order to compare them and predict which page should be the original one.

After having stored Facebook's information, the second step is to query the new index and find which local Facebook page may match with the single company. This step in very important because it shifts the problem from a "one-vs-all" classification to a "one-vs-few" classification. The query selects a small subset that may match with the single company and makes possible to compare the company with only 20-30 pages/users instead of all the dataset. Selected the company's pages/users, a metrics script has to define some feature of the pages, giving a list of n-uples representing how similar the page/user data is to the company one.

The last step is to train a classifier with the features computed in the previous step and then classify all the pages. At the end of the classification each company with a Facebook page should have the page linked by the classifier, with a percentage of how similar it is to that company's data. The classifier algorithm chosen is random forest. The choice was made for time reasons, for avoiding overfitting problems and because random forest is easier to be configured in an affordable way and amount of time.

One big problem is that some company name are not equal or contained in their Facebook page because the society name is only used for legal purpose and the real name is actually different. In that case it's impossible to find the page with Facebook's GraphAPI and the result cannot be contained in the resulting set of the classifier.

4 Solution

4.1 Solution

The following paragraph contains a description of how the solution to the problem was developed and why certain choices were made.

4.1.1 Getting Possible Facebook Pages

The first module of the project has the task to get company names contained in the Atoka index and query Facebook Graph via Facebook GraphAPI in order to get a list of possible compatible pages.

For each company contained in Atoka index, the module generate a list of different possible alternatives to the legal name. This approach extends the time spent making multiple queries but makes sure to have the most number of possible Facebook Pages/Users per company.

The first correction is to add the cleaned legal name. The module cleans the name from company's legal form so as to make a reasonable request to GraphAPI that returns a bunch of pages that match with the given query.

An example of a cleaned name is:

[¿=stealth, node distance=3.1cm, database/.style= cylinder, shape border rotate=90, aspect=0.25, draw] [database] (db1) at (0,0) Atoka Index; [database,right of=db1] (db2) Facebook Graph; [database,right of=db2] (db3) Atoka Facebook; [-¿,blue!50] (db1) – ++(0,1.5) – (($db2$) + (0, 1.5)) node[black,midway,above,font=]Graph API POST Request – (db2) ; [-¿,blue!50] (db2) – ++(0,-1) – (($db3$) – (0, 1)) node[black, midway,below,font=]Graph API JSON Response – (db3) ;

Figura 4.1: Process of data acquisition from Facebook GraphAPI

"*Spaziodati S.R.L*" ⇒ "*Spaziodati*"

A second cleaned name to add is the one with patterns associated to company legal forms removed:

"*Bar Roma di Rossi Carlo e figli*" ⇒ "*Bar Roma*"

Other names added are the ones with all possible legal forms denomination alternatives. Atoka index in fact contains companies' name with their own legal form sign. The project used a dedicated function of Dandelion in order to generate all the options.

An example of one a new alternative is:

"*ENI S.P.A*" ⇒ "*ENI Societa per Azioni*"

or can simply be:

"*Enel S.P.A*" ⇒ "*Enel SPA*"

Another step is to change the legal name with the many possible alternatives names contained in Atoka index.

A plausible example is:

"*CervedGroup S.P.A*" ⇒ "*CG SPA*"

Even combinations of the alternatives are made:

"*CervedGroup S.P.A*" ⇒ "*CG Societa per Azioni*"

An important thing made achievable by Atoka data is to add to the list signs' names. Signs' names are names contained in the signs outside the companies and in many cases represent the real name of the Facebook Page/User. This fact happen because it's not unusual to have different local stores with different names that belong to a single holding company that will have nothing to do with the single marketing strategies and social activities.

That cases are usual initials, in the form of:

"*ABC S.N.C*" ⇒ "*Ristorante da Luigi*"

As briefly explained before, every name is added to a list and every company will have a query associated at each name in that list. This makes sure that every possible page name is searched inside Facebook Graph and stored in the new elasticsearch index. We found that is far better to call 10 queries for each company and find a reasonable amout of pages that may match rather than populate the index in less time but having less possibility to find a matching page.

For using GraphAPI, Facebook requires a Developer account, verified with an individual mobile telephone number. Each account can have different applications running at the same time. The applications are identified with a text token and each of them can make at most 100M API calls per day[1]. Even if the documentation shows that limit, experimentation found that the real limit is around 600 calls per 600 seconds, per token per IP[2]. This bottleneck is overtaken creating 5 different accounts with 5 applications each, with a total of 25 applications. The module is structured in multiprocessing with one different process for each token, running simultaneously on a private server. A second trick used not to be banned from Facebook is making requests with a customized version of facepy[3] library that changes proxy continuously and simulates request all around the world. Every request is made after a random delay timeout, chosen to simulate real world's typical application calls.

A batch request makes possible to query the API 50 times at one, optimizing the latency created by http's connection opening and closing. Each query is composed by two different groups of graph Api subqueries, one group for user profiles and one group for Facebook pages. All the two groups of requests put together the names generated in the previous step. Each request return at most 30 entities, the limit is chosen not to have a too large amount of useless data: most of the time the right company page/user comes in the first 20-30 results.

4.1.2 Cleaning and Normalizing Data

Before saving all the pages and user results in the new elasticsearch index, the data is cleaned and normalized. This step is obligatory because pages and users have different fields and because very frequently data exists but can't be found in the right place. This happens because people seldom set up their company page/user profile without taking care of adding the right thing in the right place. For example, in a lot of different pages, the field *"phone"* or *"email"* is empty but you can find one

[1] https://developers.facebook.com/policythingstoknow

[2] http://stackoverflow.com/questions/8713241/whats-the-facebooks-graph-api-call-limit

[3] https://github.com/jgorset/facepy

Type	Attributes
Page	id, name, link, about, email, location, likes members, description, category, contact_address, general_info, general_manager, mission, phone, website, cover
User	id, name, link, about, email, bio, work, location, description, address, hometown, website, cover

Figura 4.2: List of attributes exposed by Facebook for Pages and Users through Graph API.

or both of them in the *"about"* field, that was originally thought to contain a brief description of the page and what it is about.

To make sure to have all the data cleaned and normalized the module combines the fields in a single structure. After having collected everything in one place, the module checks with a proper regex if possible domains and emails are contained in the structure and annotates the structure with ontologies extracted with Dandelion. Dandelion extract persons, telephone numbers, places, addresses and entities (like "data mining", "agriculture", "automobiles"). Telephone numbers and addresses are automatically normalized by Dandelion and can be directly checked with the same attributes contained in the Atoka index.

The only field that is not included in the structure is the location information, that is analyzed on its own to avoid problem of ambiguity finding, for example, people inside the structure.

Ex : "[..] $Via\ Marco\ Polo$, 13 [...] $\Rightarrow \{\}$

After having cleaned and normalized the information, data is stored in the new elasticsearch index. The primary key chosen to identify the tuples is the real Facebook ID: this choice becomes handy because it make sure that one page/user is store only once and that the index is redundancy free and consistent.

4.1.3 Generating Similarity Scores

After the data is normalized and stored we need to generate the similarity score between the company and the possible Facebook pages. This is a list of how the scores are gives for each property:

- WEBSITE Check if the sites are present: 0.5 pts if not. Matching of the normalized sites: 1pt if positive, 0 if negative. Sites normalization works removing from the string all the unnecessary parts, such as "hhtp://", "https://", "www.", we then match the two strings fully

- EMAIL The results are in form of tuple containing the resulting scores. First we check the presence of the email addresses: (0.5, 0.5) pts if not present. Next we match the various components of the input email with the company registry. The email is split in it's three main components: username, domain and ext, and for the companies we use data such as complete name, city and province code. For every input we produce the rankings with the company registry, using username first and domain in another run. Each run we clean up the input data, we fetch the province's full name and create a token list that are in common with the email and the processed data. Next pass is to check if every input token exceedes a similarity threshold, set at 75% of the total length. Then the suitable ones are divided into two categories, common matches and uncommon matches. The token total score is calculated, adding up all the 'common matches'. Same for the 'uncommon matches'. Finally we produce the final score, choosing the highest couple

  $\left(\frac{scoreCommon}{length(username)}, \frac{scoreUncommon}{length(username)}\right)$

  The same thing is done for domain:

  $\left(\frac{scoreCommon}{length(domain)}, \frac{scoreUncommon}{length(domain)}\right)$

- PHONE Check if the phone numbers are present: 0.5 pts if not. Matching of the inputs filtered from the empty ones: 1 pt if positive, 0 if negative.

- NAME Check for the presence of a description or a name in the facebook input: 0.5 pt if not present. We firstly checked in the description if there was at least one of the alternative names of the company: 1 pt if at least a match is present, 0 otherwise. We decided to optimize the compare in order to avoid to give a too high rate for too usual name. In fact it happens frequently to have situation where different companies have a very similar name:

  $Pizza\ Vesuvio\ di\ Nino\ Coppola \simeq Pizza\ Vesuvio$

  the formula chosen to overtake this problem is:
  $\frac{\#tokenAtoka \cap \#tokenFacebook}{\#tokenAtoka \cup \#tokenFacebook}$

- ADDRESS For this section we calculate multiple scores for each part that the Facebook input address is consisting of.

  - STREET Check for the street presence: 0.5 pts if not. Matching using the subdivision of the input streets in tokens, and then analyzing their set intersection using the formula:
    $\frac{\#tokenAtoka \cap \#tokenFacebook}{\#tokenAtoka \cup \#tokenFacebook}$
  - HOUSE NUMBER Exact match between the input numbers: 1 pt if positive, 0 pts if negative or The function can also calculate a score determined by the number distance of the 2 inputs, using the formula:
    $1.5^{-|house\_number\_1 - house\_number\_2|}$
  - CITY Check for the city presence: 0.5 pts if not Exact match between the two cities: 1 pt if positive, 0 if negative.
  - PROVINCE Check the presence of the province: 0.5 pts if not Exact match between the provinces 1 pt if positive, 0 otherwise
  - ZIP CODE Check the presence of the zip code: 0.5 pts if not. Exact match between the two zip codes: 1 pt if positive, 0 otherwise
  - TOTAL Results are in form of tuples having 5 scores each. Firstly we control the presence of all the above scores, and put a 0.5 if they are absent (or 0 in case of the house number) Each tuple is of the form:
    ($ScoreStreet$, $ScoreHouseNumber$, $ScoreCity$,
    $ScoreProvince$, $ScoreZipCode$)
    If no score is present, the resulting tuple is:
    (0.5, 0, 0.5, 0.5, 0.5)
    Otherwise, we choose the best scoring tuple, from all the comparison of all the Facebook addresses with the Atoka input.

- ENTITIES Score given by the intersection of the input data entities using the following formula:
  $\frac{EntitiesAtoka \cap EntitiesFacebook}{EntitiesAtoka}$

4.1.4 Training Random Forest Classifier

After the process of score assignment a subset of results is used to to train the classifier. The algorithm chosen for the machine learning classification is random forest.

A random forest is a type of classification algorithm that uses a multitude of decision trees for finding the resulting class of an input entity. The output is defined by the mode of the classes of all the decision trees that operate each one on a subset of the total of the attributes given to the classifier. The main principle is that using a lot of the so called "weak learners" (the single decision tree) on all the same target, we can obtain a "strong learner" (the random forest) made by their single valuations. Basing the conclusion on the mode of the single outputs has a drawback, as more and more trees are used the error rate will rise but contrariwise, as we lower them, the strength of the single element goes down: a difficult task is finding the number m of elements.

Random Forest is generally faster than the other types of classification algorithms, and this is one of the reasons because it's the best algorithm found for this problem[**?**]. In a slightly different

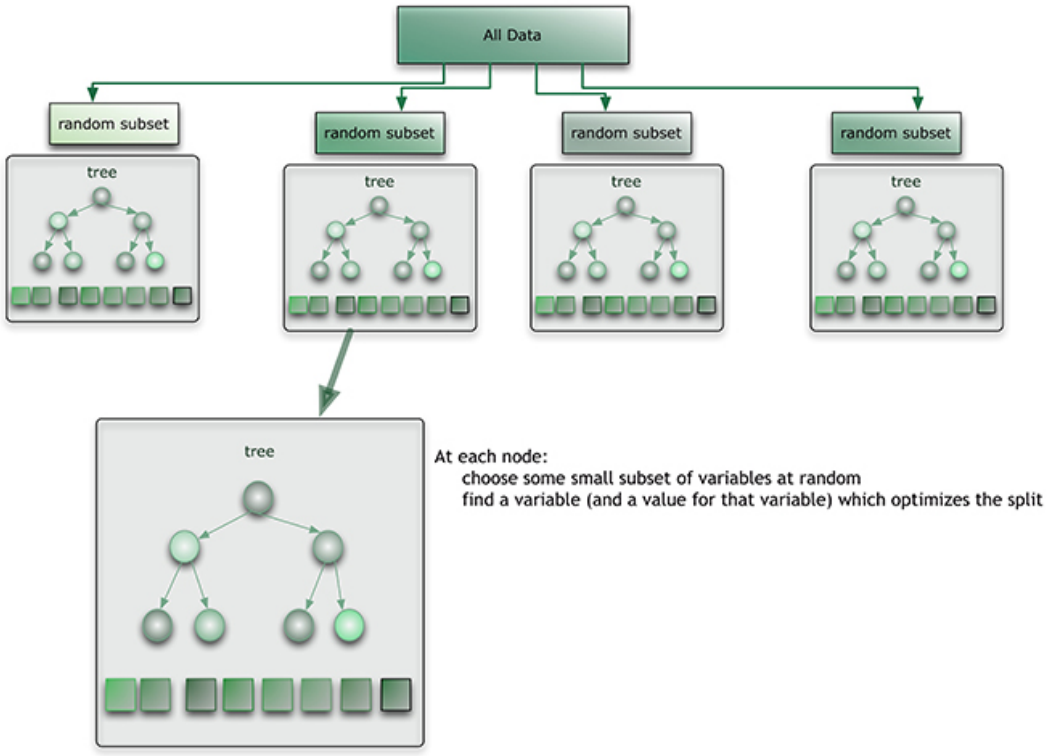

Figura 4.3: Example of how a Random Forest is composed

problem, the classification of websites instead of Facebook pages/users, SVM was implemented but the running time was tens of times longer than Random Forest and the results were even worse. This shows the second reason of why Random Forest seems to be better than SVM or other approaches: it's easier and faster to set, to regulate and to train. Other important facts that showed Random Forest Classification Algorithm as the best for this job are that it's very good avoiding overfitting, that it's solid with outliers and that it works fine with partial and dirty data.

The first step when somebody is working with a machine learning algorithm is finding a useful training subset of the data. The main training subset is composed by 60% of the data, splitted randomly from the main dataset. Using the Random Forest Algorithm it is not necessary to define a custom cross validation system because it already have its own method to estimate the out-of-bag error. Even if the algorithm comes with this "simplification" we chose not to use it and to implement a cross validation. This choice was driven by the fact that the dataset is really skewed, with only few positive examples and few association between companies contained in Atoka index and theirs Facebook pages. The approach used is stratified three fold cross validation: the training set is sampled in the three fold, making sure to maintain the same positive/negative percentage composition, and for three time the random forest is trained with one fold and tested on the other two. At the end, the trained classifier is tested with the remaining 40% of the data.

In the training step, instead of giving parameters as possible values, the approach used was to give them as distribution with the Randomize Grid Search method. Randomize Grid Search randomly samples the distribution, then tests the different parameters and takes the better ones. The method uses 100 samples for the training dataset. This method works fine with this problem because it gives decent results if the distribution is uniform (as it is) and works in a reasonable amount of time.

The results with the first training set were not very good. This problem was caused by the fact that ATECO n.56's companies only had 6,000 pages/users classified and because many of that pages were false positive and many other pages that were missing in the initial index were found right in the training set. To avoid this huge problem we opened a story on CrowdFlower[4], asking people to verify if the pages associated by the classfier were right, wrong. This process help us redesigning the training set without false positive and true negative. This may seems a conscious modification of the training dataset to have "better final result" but, in reality, it is not overfitting because it only fix error that should not be in the first dataset. This solution was the only adoptable because the only platform that surely knows if a page is really about a company is Facebook itself.

5 conclusions

The final data presents a precision that start from 0.9 and grow to 1 while the recall, starting above at 0.95, after a threshold of 0.75 drastically fall down, getting to zero very fast.
The threshold chosen is 0.73, that assures a precision of 95% and a recall of 90%. This means that we give a wrong page one time over twenty and that we loose a good page one time over ten.

We chose a really high precision rather than a higher recall because in a business product like Atoka it is far more important not to give wrong information instead of losing a result that sometimes could be available.

We started with 362154 companies with ATECO n.56 collected in Atoka index. The Facebook pages connected to that companies, before the project was only 6355. The number of pages collected in the first Facebook index was 1,321,634.

After this work, with the threshold chosen at 0.73, the Facebook pages/users matched with Atoka companies are 63172. At a lower precision with a lot higher recall, founded with a threshold of 0.5, the number of matched pages amount at 79210.

This project help Atoka to classify Facebook pages/user for a total of 10 times more than before the project. The percentage of companies with ATECO n.56 that now are matched with a Facebook

[4]http://www.crowdflower.com/

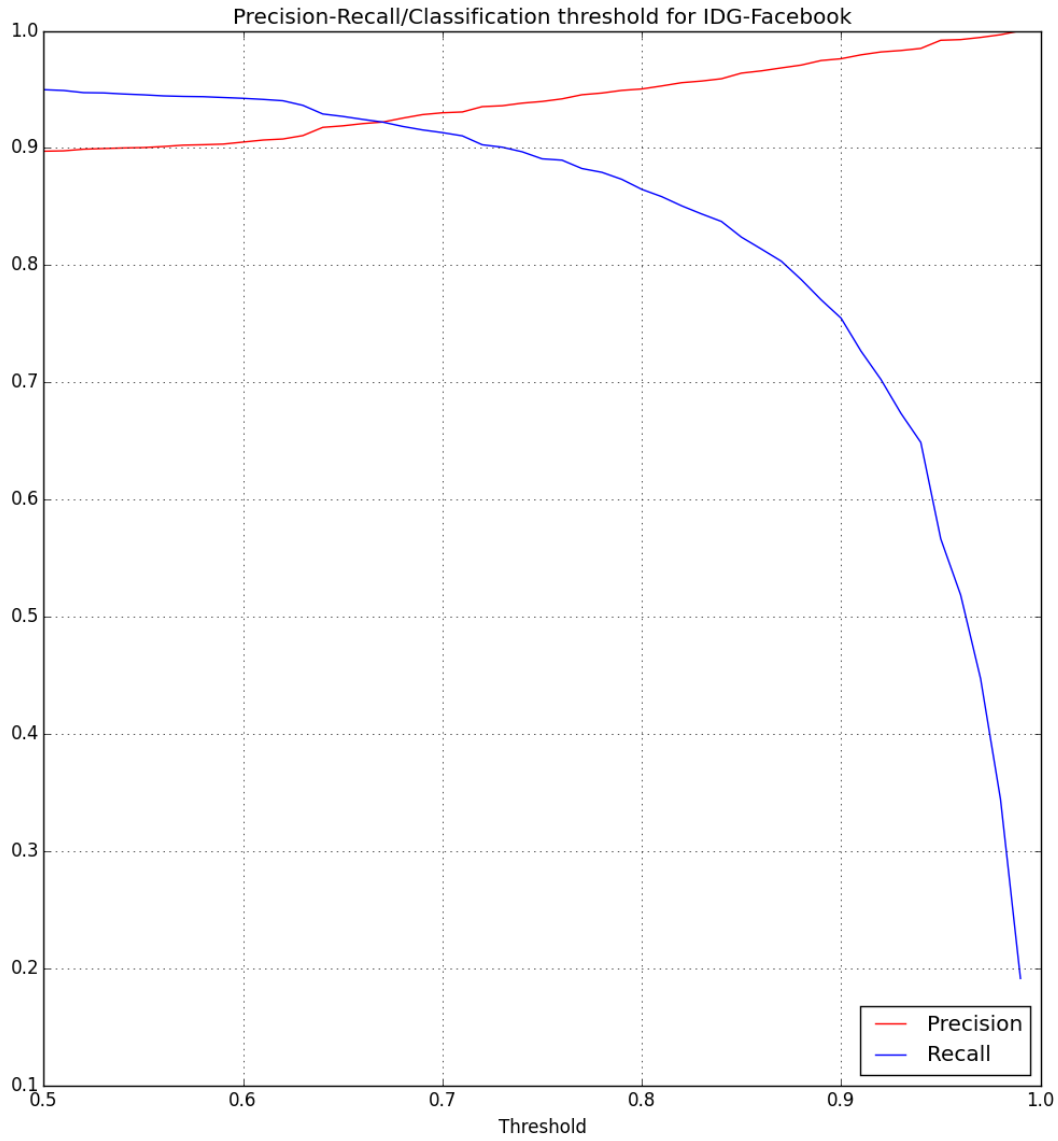

Figura 5.1: Precision-Recall/Classification Threshold Plot

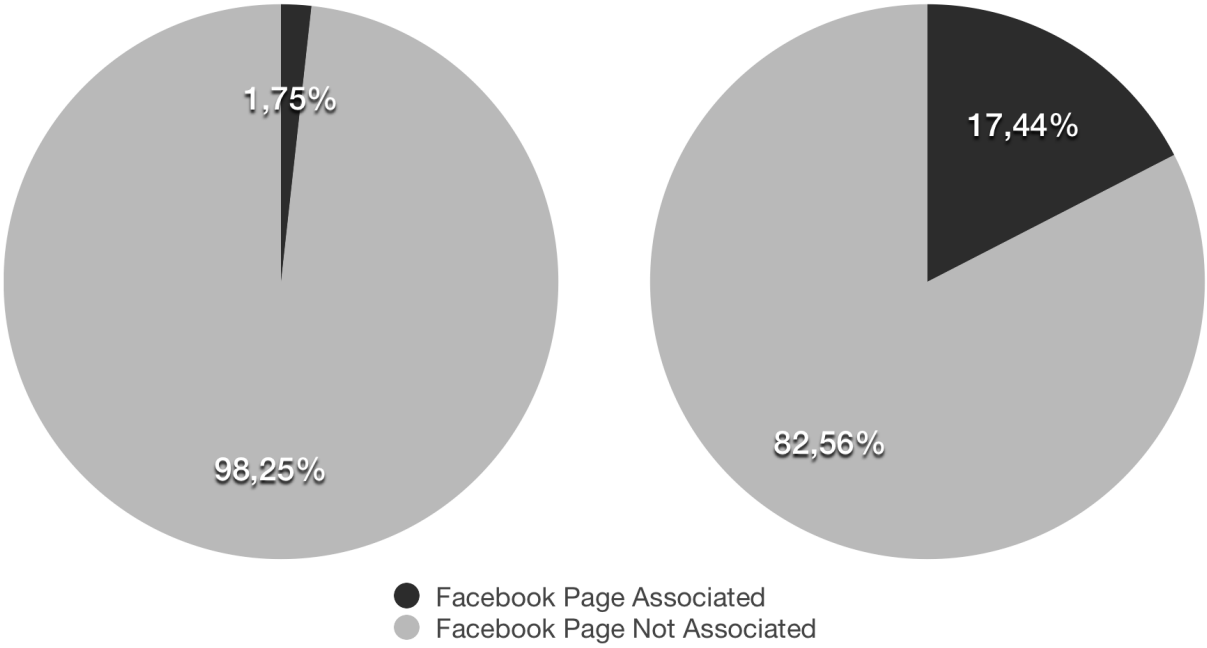

Figura 5.2: Percentage of Facebook Pages matched before and after the project

has grown to 17.44%.

6 Related Work

7 Implementation

7.0.1 GraphAPI

The GraphAPI can be queried with a single POST request. Here's the code to compute the batch request:

```
[{'method': 'GET',
    'relative_url': 'search?q={0}
    &type=page&locale=it_IT&limit=50
    &fields=id,name,link,about,email,
    location,description,category,
    contact_address,general_info,
    general_manager,mission,phone,
    website,members,cover,likes'
    .format(
        quote(unicode.encode(company_name,
        encoding='utf-8')))
```

```
} for company_name in company_list
] + [
{'method': 'GET',
    'relative_url':'search?q={0}
    &type=user&locale=it_IT&limit=50
    &fields=id,name,link,about,email,
    bio,work,location,description,
    address,hometown,website,cover'
    .format(
        quote(unicode.encode(company_name,
        encoding='utf-8')))
} for company_name in company_list
]
```

7.0.2 Dandelion API

Dandelion API is a Semantic Text Analytics as a service product developed by Spaziodati. Dandelion is a product that implement context intelligence: it extracts meaning from an unstructured text and put it in context, obtaining actionable data. Instead of classic NLP technologies, Dandelion API leverages its underlying Knowledge Graph, without relying on traditional NLP pipelines. This makes it faster, more scalable, easier to customize and natively language independent. Dandelion API works well even on short and malformed texts in English, French, German, Italian and Portuguese. The main APIs available on Dandelion are:

- ENTITY EXTRACTION: find mentions of places, persons, brands and events in documents and social media. Easily get additional data about the entities.
- TEXT & CONTENT CLASSIFICATION: classify multilingual text into standard, pre-defined taxonomies or build your own custom classification scheme in minutes.
- SENTIMENT ANALYSIS: identify whether the expressed opinion in short texts (like product reviews) is positive, negative, or neutral.
- KEYWORDS / CONCEPTS EXTRACTION: automatically identify important, contextually relevant, concepts and key-phrases in articles and social media posts.
- SEMANTIC SIMILARITY: compare two texts and compute their syntactic and semantic similarity. Understand when two texts are about the same subject.
- ARTICLE EXTRACTION: extract clean text article from newspapers, blogs and other websites. Remove boilerplate and advertising and get the article full text and images.

7.0.3 Atoka

Atoka is a lead generation business-to-business tool that helps companies to find new customers, receive news and updates from them and to generate prospect lists with few clicks. Thanks to the partership with Cerved Group, an Italian rating agency, and the usage of semantic analysis algorithms, Atoka offers affordable information about more than 6 millions Italian companies and economic subjects. Atoka provide addresses, contacts, websites, social networks links, management team composition, corporate structure, economic and financial data. Atoka also enables new search options that overtake tradictional filters (like ATECO code), enabling the customer to make more precise and articulated queries.

7.0.4 Scikit-learn

Scikit-learn (formerly scikits.learn) is an open source machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

8 Experiment