CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Entity extraction for people profile matching

Master's Thesis

ALEXANDRU-DAN TOMESCU

Department of Computer Science
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2016

Entity Extraction for People Profile Matching
ALEXANDRU-DAN TOMESCU

Examiner: GRAHAM KEMP
Supervisor: FREDRIK JOHANSSON
Advisor: TRAIAN REBEDEA

Chalmers University of Technology and University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Goteborg
Sweden
Telephone: + 46 (0)31-772 1000

Department of Computer Science and Engineering Gothenburg, Sweden, June 2016.

Abstract

With the rise in social media platform usage, the average number of people profiles has increased as well. The fact that people have social media profiles on multiple platforms reveals the interesting problem of matching them in order to aggregate all the profiles in one. As a significant part of these profiles are made of unstructured text, extracting labeled data using an entity extraction system can lead to an increase in precision and recall.
This thesis documents the effects a bootstrapped pattern learning approach can have on a profile matching system. The entity extraction system is trained in a distributed manner on a big amount of data, in order to generate as many quality patterns as possible.

Keywords: entity, extraction, pattern, bootstrapped, distributed, profile, matching.

Acknowledgements

I would like to express my gratitude towards the advisor of this thesis, Traian Rebedea for the continuous feedback and support during the development, as well as the team at Peoplegraph for the help and advice. Also I would like to thank Fredrik Johansson and Graham Kemp for their thorough feedback, support and interest in the subject.

Alexandru-Dan Tomescu, Gothenburg, June 2016

Contents

List of Figures

List of Tables

1

Introduction

1.1 Introduction to People Profile Matching

There is an increasing amount of semi-structured data (a form of data that does not have rigorous formal structure but does contain labels or tags) with an important portion consisting of social media or professional profiles. Most people have a number of public online profiles on different platforms containing personal information such as name, address, education, employment history etc. Because all this information is scattered on different products, the task of searching for people is sometimes tedious and doesn't yield the best results.

Another problem is the fact that searching for a specific person just by name can result in erred results: using Google Search to look for the name of the author of this project, Alex Tomescu, yields others as well (most notably one other person with the same name that has achieved some amount of fame). Because our online identities are scattered on multiple platforms, the task of identifying all the public data about a specific person can be time consuming.

One solution for the people search task is to gather all the profiles in one place and perform the search on that index.

Having only an index with all the data would not be of much help when looking for someone. For this problem, a solution seems to be creating an identity out of all the public data for each person. This involves matching a Facebook profile with a LinkedIn profile, and/or a Google+ profile and so on.

Even though the information is usually quite similar between the platforms, matching just by name and structured data sometimes is not enough. Some of the entities which help disambiguate the relationship between two profiles are not in the structured data, but in the unstructured part of the profiles.

1.2 Introduction to Entity Extraction

Entity extraction (also known as Named Entity Recognition - NER, or entity identification) is the task of extracting words from an input text and also classifying them in a predefined set of categories. Entity extraction is a part of the Information Extraction (IE) task and has drawn attention in 1995 and 1996 in the Message Understanding Conferences editions 6 and 7 (MUC-6[1] and MUC-7[2]). The study of this problem first started with the extraction of names [1] and with time was expanded to other categories of entities. Named entity recognition is one of the first steps to extracting information and meaning from text.

One of the possible applications can be the use in the context of the matching task presented in the previous section. By normalizing the profiles and then using entity extraction to extract the entities from the unstructured text fields, matching accuracy should be improved.

Entity extraction is an important topic which can also be used in other contexts than profile matching. Having metadata for the words in a corpus can lead to structured search (commercial search engines are making a big effort in this direction). Entity extraction can also help with automatic report creation and other such tasks where the meaning of words within a text would be vital.

1.3 Terminology

The extracted *entities* from profile data are words and expressions which contain information about a certain profile: the company a given person is working in/with (Google, Microsoft, The Coca Cola Company), a role within it (software engineer, chef, manager), a location (Sweden, Romania, United States), a hobby (swimming, playing football), or a school attended (Chalmers University, Gothenburg University).

The entity extraction is trained on a data set of *social network profiles*. A social network profile is created by each user when joining the respective platform, and populates it with information, depending on the goal of the platform. The data in each profile is kept in *fields*, which differ from one platform to another. Examples of profiles are provided in the Chapter 3.

An *identity* is composed of multiple social profiles belonging to the same person. Ideally, the identity contains all the social profiles of a given person, but due to lack of data or ambiguities, it can also lack some or contain some wrong ones. The *data set* is a collection of documents containing text (in this case social media people

[1] http://www.cs.nyu.edu/cs/faculty/grishman/muc6.html
[2] http://www.itl.nist.gov/iaui/894.02/related_projects/muc/proceedings/muc_7_toc.html

profiles) which is used when training or evaluating a system. Subsets of this are the *training set* and the *control set* (for which we already know the wanted output of the system applied, which we can compare against).

1.4 Scope

In this thesis we concentrate on entity extraction from social media people profiles, using a data set of profiles both for training and for evaluation. Even though the case analyzed here is narrow, the entity extraction system can be trained on any semi-structured data that has labeled fields containing examples of entities that we are looking for. After training, the entity extraction system can be used on any type of text.

1.5 Approach and Goal

There are two main goals we wish to achieve with this thesis: to see how a semi-supervised entity extraction algorithm works on social media profiles and to see what improvements the entity extraction can bring if plugged in a profile matching system.

The project started by evaluating the existing entity extraction algorithms and by selecting the data. Next, after researching the available literature on the subject, given the nature of the data, the algorithm was chosen, and implemented in a distributed manner. The algorithm was evaluated on a data set and when the best results were achieved, it was integrated into an already developed matching system.

1.6 Thesis Outline

In Chapter 2 the context in which the entity extraction and matching are performed is presented. An overview of the recent state of social networks (overlapping between social networks, the use of social networks by the average person) and the efforts of PeopleGraph, the company in collaboration with which this project was realised, are given.In Chapter 3, the data set used in entity extraction and in the matching phases are described in detail.
Next, in Chapter 4, the entity extraction system is described in detail (both infrastructure, algorithm and steps). In this chapter there is also an overview of how it was used in the profile matching system. The entity extraction system is then evaluated in Chapter 5, as well as how matching is improved. The related literature is then presented, both in terms of entity extraction and profile matching, and approaches

are compared in Chapter 6. In Chapter 7, the ethics of entity extraction and social profile matching are discussed, and in Chapter 8, conclusions are presented.

2

Context

This project is executed in collaboration with the startup PeopleGraph located in London/Bucharest, under the supervision of Dr. Traian Rebedea.

PeopleGraph's mission is to "help the right people find each other", which materializes in people search products (the task of finding people online), such as Closeround[1]. To enable high quality people search, an index containing more than 2bn profiles has been created for platforms such as Google+, LinkedIn, Facebook, AngelList, Crunchbase, Instagram, about.me and others.

One of the concepts that stands at the core of people search products is a way of aggregating multiple public profiles belonging to the same person into one identity representing the real person. The project described in this document aims to solve the problem of extracting entities from the unstructured text of a profile, with the final aim of using those entities as additional information when matching to form an entity.

While using entity extraction for improving the results of a profile matching system is an interesting problem, it is not the only one. There is an ongoing effort to structure the big amount of online data, and entity extraction is one of the basic sub-tasks involved. One possible application for entity extraction is the use in search products[2]. Even though entity extraction would be harder in this context, as queries are quite small and lack context, disambiguating by identifying entities and their classes could provide improvements in search results.Another possible application can be the extraction of events, as they are identified by a few entities: start and end date, name and location as seen in [3].

With the rise in number of social media users, a big amount of data is generated containing information on topics of interest for the general public. As this is an important indicator of commercial success in many industries (e.g. technology, music, cinema), trend analysis on social media has become an important market. Extracting company names from tweets or Facebook posts and analyzing the number of occurrences can quantify the interest generated by news articles or product launches. This is similar for music and cinema and many other markets and industries which rely on the "buzz" generated around them. The subject of extracting entities from

[1]https://closeround.com

Twitter has been discussed by Ritter et. al. in [4].

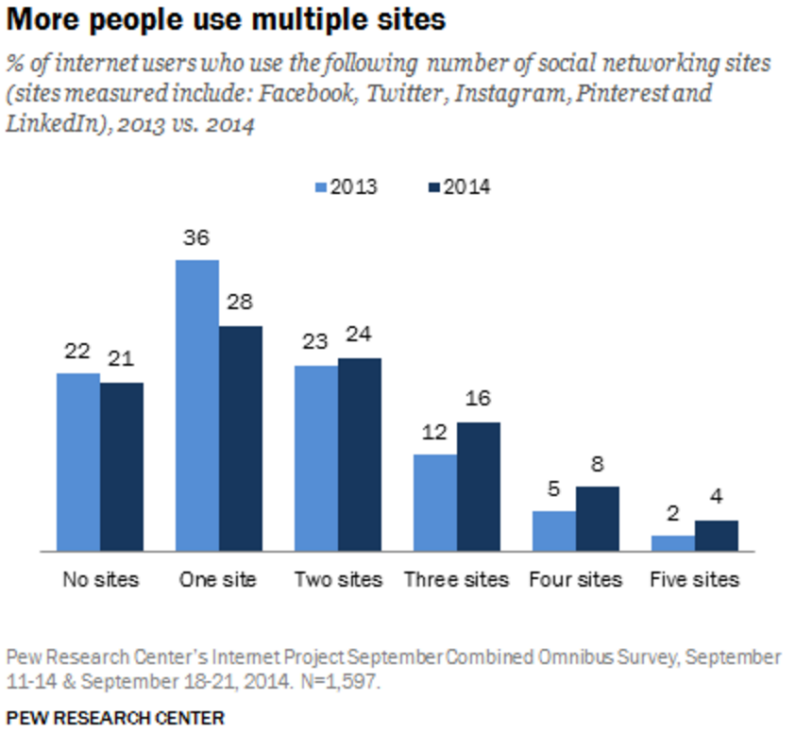

Figure 2.1: % of internet users who use Facebook, LinkedIn, Twitter, Instagram and Pinterest, 2013 vs 2014 [2]

Research made by the Pew Research Center [5] in 2014 illustrates the fact that a big chunk of Internet users have accounts on multiple social networks: 52% of users have more than one. From the table in Fig, 2.1, it is also obvious that the numbers shift towards having social profiles on more platforms. Going into more detail, the overlap in platforms in 2014 is analyzed, Facebook being the most used, and having most users in common with other platforms (especially as Instagram is owned by Facebook).

According to an article on GlobalWebIndex[6], monitoring of the usage of close to 50 regional and global social networks revealed that the average number of social media accounts for an Internet user was 5.54 in 2015 (and has likely increased during the year, as studies show that the number of social media users continues to grow[3]).

As seen in Fig.2.2 and Fig.2.3, the number of LinkedIn and Twitter users has increased over time (for Twitter it seems to be stagnating in the last few quarters).

[2]http://www.pewinternet.org/2015/01/09/frequency-of-social-media-use-2/

[3]http://www.smartinsights.com/social-media-marketing/social-media-strategy/new-global-social-media-research/

[4]http://www.statista.com/statistics/274050/quarterly-numbers-of-linkedin-members/

[5]http://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/

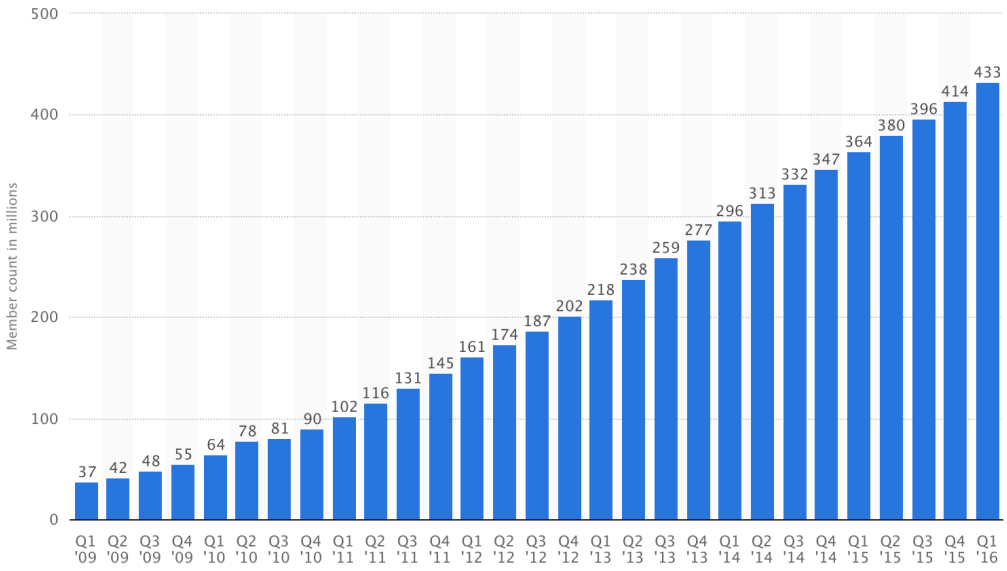

Figure 2.2: Number of LinkedIn users over time according to statista.com [4]

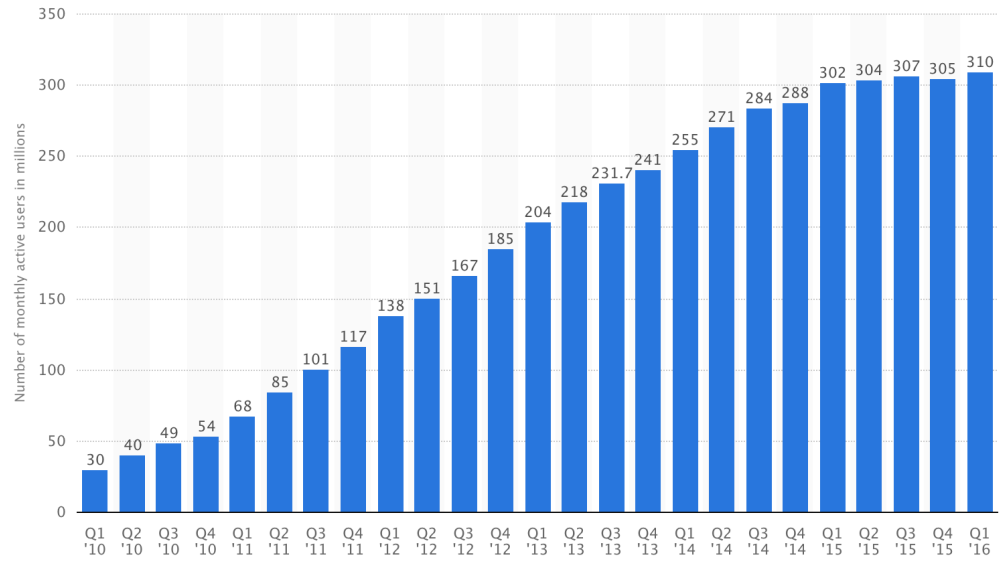

Figure 2.3: Number of Twitter users over time according to statista.com [5]

This trend is also obvious on other very popular social networks.

All these facts further strengthen the use of building public data identities in order to achieve an important part of structuring the web. Another important aspect of the social media usage is the fact that people are only active on an average of 2.82 profiles[6]. This leads to out of date profiles which might contains important information which will be hard to locate. There are numerous applications for social profile matching and information aggregation about people, depending on the goal and the domain. Applications range from recruitment products which aggregate all

[6]http://www.globalwebindex.net/blog/internet-users-have-average-of-5-social-media-accounts

the online data about a certain candidate such as skills, previous work experience and projects.

Aggregating identities can also prove useful in the case of public figures such as politicians in order to give their electorate a more thorough and adequate idea about them, or in the case of celebrities so that people can follow multiple accounts of people they admire. The idea of building a platform where politician profiles can be aggregated with the goal of informing the public for elections has generated interest such that PeopleGraph has been awarded a grant for building one by the Open Data Incubator for Europe (ODINE)[7].

[7]http://www.theguardian.com/odine-partner-zone/2016/may/19/winners-of-the-fifth-odine-call-announced

3

Dataset

In this chapter the dataset used for training and evaluating the entity extractor and the effect it has on the matching results is presented. Details such as the structure of the data, the particularities of some of the profiles available and examples, provide a clearer picture of the data used in this thesis.

The corpus used for training this system consists of more than 2 billion profiles stored on the Amazon S3 platform[1]. The profiles are normalized and contain information from platforms such as LinkedIn, Google+, Facebook etc. Through the nature of social profiles, they contain both structured and unstructured data. Structured data is represented by name, work history, location, education (in most social profiles), but also some free form of content (descriptions and headlines in LinkedIn). The structure of a LinkedIn and a Twitter profile are presented in the table in Fig.3.1

		Linkedin	Twitter
id		linkedin/alex-tomescu-445a0633	twitter/alextomescu
url		https://us.linkedin.com/in/alex-tomescu-445a0633	https://twitter.com/alextomescu
name		Alex Tomescu	Alex Tomescu
headline		Software Engineer at Wholi	Software Engineer at Wholi and former intern at Google. I study at GU in Gothenburg and I'm a big fan of Elon Musk !
description		My experience so far is made up of various internships: three at Google, one Google Summer of Code, a summer as a web developer at a small Bucharest company, and finally at the Wholi startup (with which I am also working on my Masters dissertation).	
positions	**company**	Wholi	
	role	Software Engineer	
	period	December 2015 - Present	
education	**degree**	Master's Degree, Computer Science	
	school	Gothenburg University	
	period	October 2014 - Present	
location		Bucharest	Bucharest

Table 3.1: Normalized LinkedIn and Twitter profiles

The data contains mostly profiles written in English, but there are also other languages present: Spanish, Italian, Japanese, French and others. Initially, the profiles which contained words not written in English were discarded, but that proved to be unnecessary: most languages contain patterns which predict entities like companies or schools. The algorithm extracted multiple patterns which appeared frequently

[1]https://aws.amazon.com/s3/

enough that they could be considered confident patterns: etudiante en , trabajar en .

id		aboutme/alex.dan.tomescu
url		https://about.me/alex.dan.tomescu
name		Alex Tomescu
headline		Software Engineer, Student, Wanderer
description		
positions	**company**	Wholi
	role	Software Engineer
	period	
education	**degree**	CS Masters
	school	Gothenburg University
	period	
location		Bucharest
other_profiles		linkedin/alex-tomescu-445a0633
		twitter/alextomescu

Table 3.2: Normalized about.me profile

The most valuable type of profile for this task is the LinkedIn profile because the platform is oriented toward work relationships. It contains work history and education history, and the fields can be used for extracting seed entities. Also, the description and headline fields are usually populated with relevant information that may contain work and education history. The fact that well written and complete LinkedIn profiles represent a plus in the recruitment world, the data is sometimes more relevant and complete compared to other platforms.

Other profiles have more unstructured data which makes them harder to match to others (Twitter). In Twitter, other than the name, the location, and some follower count information, there are not many structured fields that can be used in matching, but in the headline(bio) field, people sometimes give enough information about themselves, the company they work in, or where they study. This is where the entity extraction algorithm comes in: by extracting entities from the unstructured parts of the text, there is more information to decide on in the matching process.

In the dataset there is also a portion of profiles which cannot be properly used because they do not have much information. Some Google+ profiles do not contain either structured information about work history or education history, or unstructured description or headline fields. These kinds of profiles are not taken into account.

Profile from photo sharing social networks such as Flickr are hard to match because of the lack of information, as people create an identity mostly through media content

they created or redistributed. Instagram profiles contain a headline which could be used for entity extraction, but the content is usually more non-conformist and does not provide information about work history or education. The about.me platform contains identities voluntarily created by people by providing basic information and links to their profiles on social network platforms (Fig.3.2). This comes in useful when creating the control set for evaluating the impact the entity extraction has on the matching profiles. The control set contains about.me profiles and the profiles the identity contains.

For the purpose of training the entity extractor, a dataset of approx. 200 million profiles are used. The entity extractor is then evaluated using a variable number profiles ranging from 30 million to 150 million profiles. The evaluation set contains different profiles from the training set.

4

Procedure

4.1 Entity Extraction

Entity extraction has been implemented in a number of ways. For this project, we have started from a semi-supervised entity extraction method presented in [7] and it is based on bootstrapping the process. In this section the implementation of the system is presented: in subsection 1 the infrastructure particularities are detailed and in subsection 2 the needed details about the Spark framework[8] are given. In the next subsections the main steps of the entity extraction training and running are presented: in subsection 3 the notion of seed entities is given and placed in the context of the system, in subsection 4 the pattern generation step is presented, and then in the final subsection, the new entity extraction process is detailed.

Bootstrapping in this context refers to the use of a predefined list of entities – known as seed entities – in order to be able to generate patterns, which are then used to generate new entities. The seed entities bootstrap the cycle.

The central abstractions and notions in the entity extraction method chosen in this project are (i) relations, (ii) seed entities and (iii) patterns.

Relations describe the type of an entity and how they relate to the context. The relations considered in this project are particular to the structure of the social network profiles and the task of matching the profiles: the company relation, the school relation, the role relation and the location relation. They are chosen based on the amount of information they can provide about a profile and on the availability of structured fields in profiles containing entities which we are sure belong to a certain relation.

The entities extracted from the structured fields (company, school, role and location fields) are considered seed entities for their respective relations. Seed entities are used for identifying patterns, which in turn identify new entities.

Patterns are composed of a sequence of words that are usually used together with an entity. When a pattern is identified, there is a high chance that the words after or before it (depending on their type) form an entity.There are two types of patterns:

- LEFT – e.g. work at ___company___, study at ___school___
- RIGHT – e.g. ___company___ is a startup, ___role___ at a company

For company and school relations, LEFT patterns are predominant, while for the role relation, RIGHT patterns are important.

The seed entities are identified in the input text, and based on the words next to it, patterns are generated. The patterns are evaluated to extract the confident ones, which are in turn used to identify new entities. This cycle can be repeated multiple times until no new entities are extracted (in the implementation described here, only one cycle is applied).

In order to take advantage of the big amount of data available, a distributed implementation is created. Because of the nature of the task (a lot of data processing in steps which are not entirely dependent on each other), the MapReduce[9] programming model seems like a good fit. For this the Spark framework [8] was used which offers a high level Java abstraction for MapReduce. This model applies well to the algorithm at hand, being translated to a number of map and reduce (or aggregate) steps, described in Fig.4.1

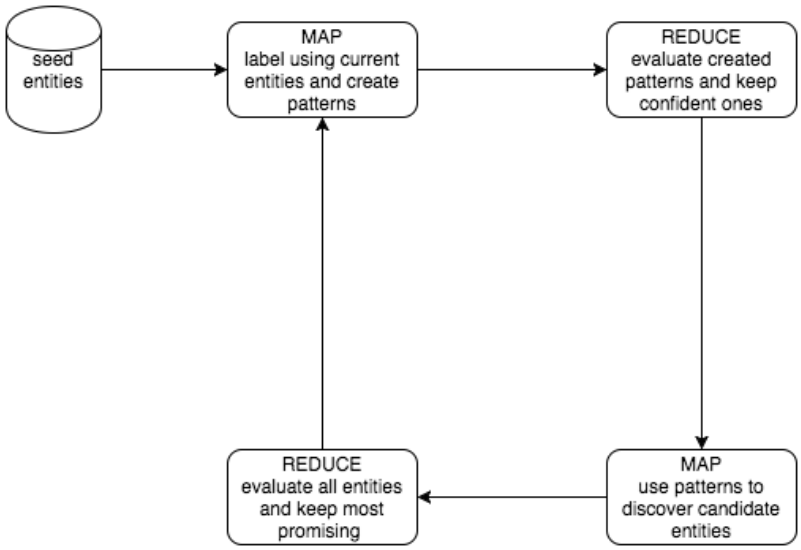

Figure 4.1: Proposed implementation for the entity/relation extraction algorithm

In this implementation companies and schools are represented by the Relation class which they extend: CompanyRelation, SchoolRelation, RoleRelation and LocationRelation. Relation classes contain relation specific information such as a list of seed entities, minimum thresholds for patterns and entities. Patterns belong to one of the relations and the entities extracted by them are tied to the same relation.

4.1.1 Infrastructure

The project is deployed on a cluster of memory intensive nodes in the Amazon Elastic Compute Cloud (Amazon EC2). There is one master node (the driver) and a number of slave nodes. The driver is not memory or CPU intensive, as it only orchestrates the computation, starting different jobs on the slave nodes. The outputs of map steps are kept in JavaRDD classes which are a high level abstraction of distributed data kept in Hadoop HDFS.

The cluster on Amazon EC2 was created and managed (Spark installation, Hadoop installation and other needed plugins) using the spark-ec2 script provided with the Spark library.

For testing during development and for evaluation, Amazon spot instances were used. Because the implemented entity extraction algorithm is very memory intensive, r3 instances were used. Not only are they memory oriented, but they also have SSD storage, which is useful because Spark has the option of storing overhead data to storage and bringing it into memory only when needed. When running the system with an input lower than 50 million profiles, xlarge instances (4 cores, 30.5 GIB memory and 1 x 80 GIB SSD storage) were used. For bigger input sizes, 2xlarge machines (8 cores, 61 GIB memory and 1 x 160 GIB SSD storage) were used.

The deployment was done using the spark-submit script also available with the Spark library.

During runtime, Spark exposes detailed information about the steps taken (default on port 8080 of the driver). As seen in Fig. 4.2, information on the progress on each action and transformation is provided. Each transformation is divided into tasks which are then distributed on all the machines, and the Spark UI also provides information on them (duration, the machine on which they are processed, status). The stdout and stderr logs can also be accessed for each of the machines, and a history of the Spark applications is kept.

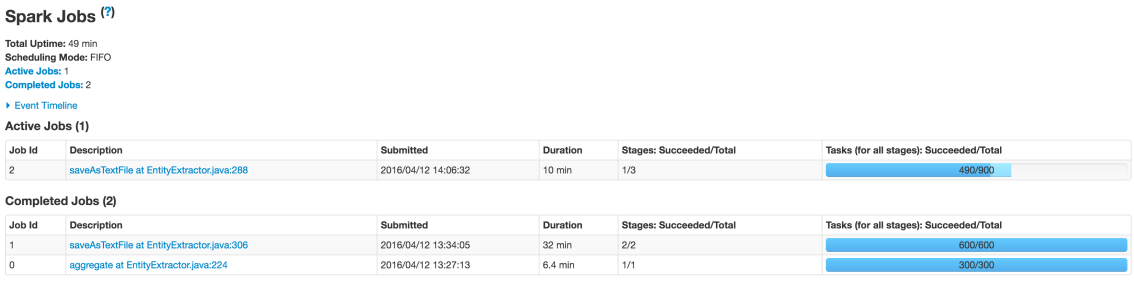

Figure 4.2: Spark UI

To monitor the load on the cluster, the Ganglia plugin was used. Detailed information on the usage of each machine in the cluster is available. In Fig. 4.3 the load_one metric is presented (the load average over one minute - the number of threads that are runnable and queued while waiting for CPU resources) as well as

overall memory, CPU and network used in the last hour. Monitoring the cluster information gives a good starting point when something goes wrong with the Spark application (many problems are caused by the use of all the available memory).

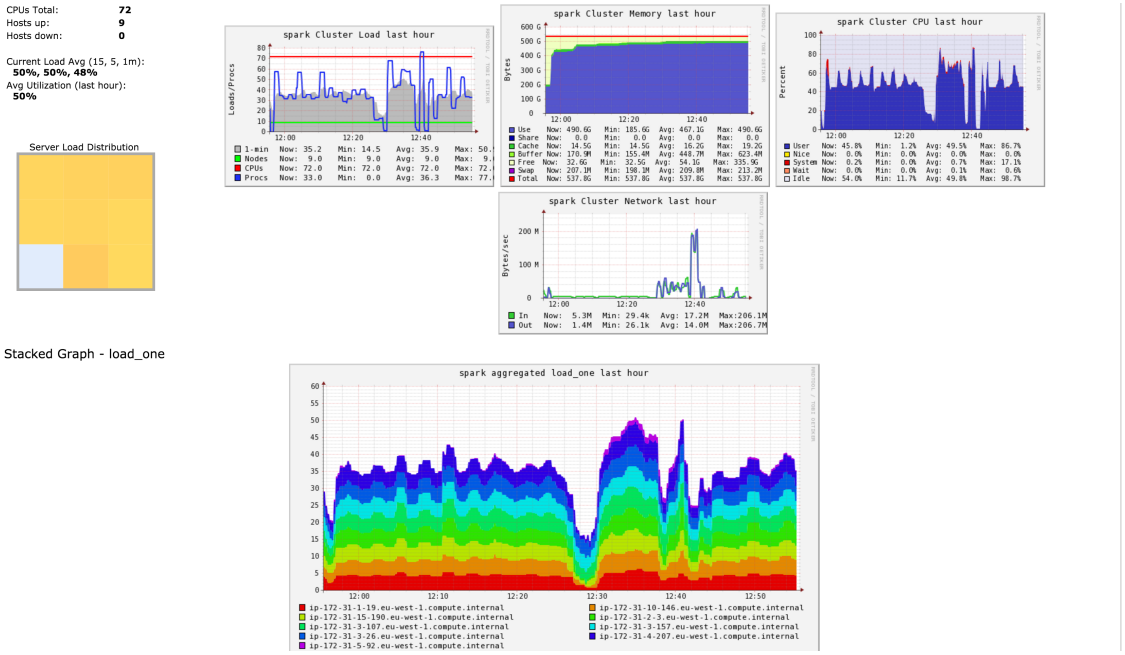

Figure 4.3: Cluster monitoring with Ganglia

4.1.2 Cluster Computing using Spark

Once the Spark application is deployed and initialized, a SparkContext is created which sets up the cluster application by granting the driver access to the workers through a resource manager (Apache YARN[1], or in this case, Spark's own manager). The program's resources are distributed to the workers and the abstraction of resilient distributed data set (RDD) is used to control the resources from the driver (JavaRDD in this case).

To process the data distributed on the cluster, the Spark framework provides two types of operations which can be performed on RDDs: transformations and actions. Transformations are operations which compute a new RDD from the old one, so the data is kept distributed on all the workers. Transformations include operations such as map (apply the provided function on each of the elements of the RDD), flatMap (a variation of map which can return a list of results for each element of the data set) or filter (the resulting data set contains only the elements for which the provided function return true). Actions are operations which give the data to the driver: reduce (aggregates the data set to the driver using a provided function), collect (aggregates all the data from the workers to the driver) or count (counts all

[1]http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

the elements in an RDD). In order to create the first RDD, data can be paralellized using the parallelize Spark operation. The effects of Spark operations are also illustrated in Fig.4.4.

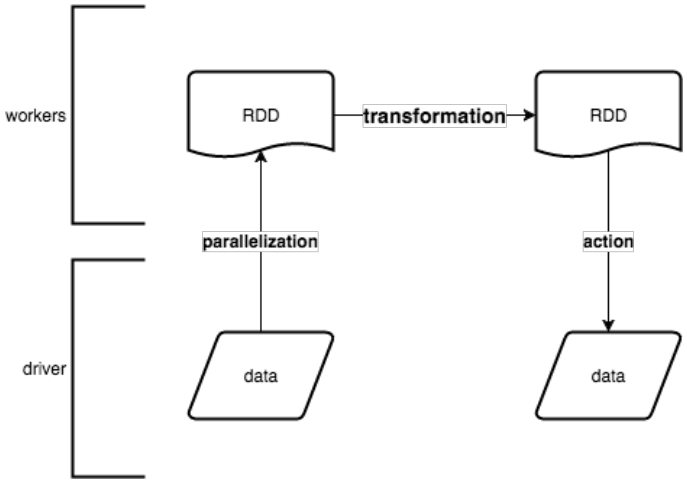

Figure 4.4: Spark operations

When working with Spark for processing a big amount of data it is always good to use transformations as often as possible, and just aggregate data when it's necessary. Aggregating the data is a costly operation which involves I/O and it can cause problems by filling all the memory of the driver.

4.1.3 Seed Entities

Seed entities are used to bootstrap the process by identifying patterns. A list of seed entities can be created either by extraction from a dictionary or by extraction from structured data. The latter has the potential of creating a bigger list of entities which increases the quality of the extracted patterns.

Due to the structure of the project's input corpus, the seed entities were extracted from the profiles which had populated fields containing work history and education history. Most of the data was extracted from LinkedIn and AngelList profiles.

The description and headline fields are used as input for pattern generation.

Fig. 4.5 gives a clear example of how a profile is used in training. The seed entity from the education field is extracted, and when it is identified in the description, the LEFT pattern "study at ___school___" is extracted.

In the training phase the data is loaded from Amazon S3 and mapped from JSON to Profile objects using the Google GSON library. By applying a map action to the loaded profiles a JavaRDD is created and distributed along the nodes.

Based on the generated Profile objects, company and school names are extracted.

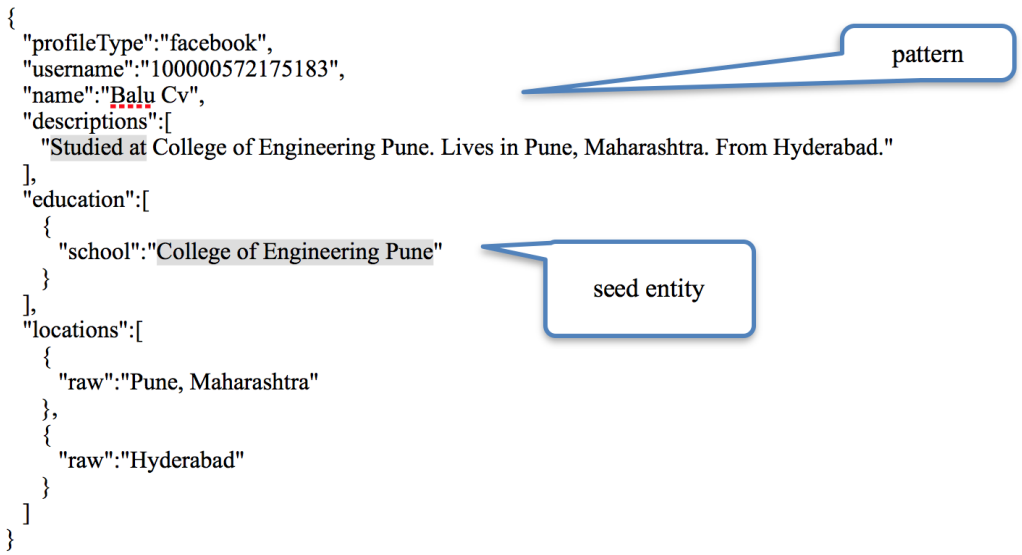

Figure 4.5: JSON profile with pattern and seed entity

4.1.4 Pattern Generation

In a preprocessing step, all the entities found in the structured part of the profile are replaced in the description and headline properties. In this step, Stanford CoreNLP [3] is also used in order to tokenize and lemmatize each word. This results in Token objects which contain the initial form of the word, the lemma and a boolean specifying whether or not the word was an entity.

As the tokenized input set is quite big and the most expensive to compute, it is kept in cache, using the Spark Framework persist method, configured to use both memory and storage (MEMORY_AND_STORAGE is set as caching method). By doing this, the tokenized input is only computed once (not every time it is needed) which yields a significant time improvement. If the data handled is bigger than the memory capacity of the machine it will be saved in the storage until needed.

Pattern generation is done in a PairFlatMap stage, in which the program iterates through all the tokens from the headline and description of a profile. When an entity token is identified, patterns are created by taking into account the words to the left and to the right of the identity: the ngrams to the left of the seed entity are considered as LEFT patterns, and the ngrams to the right of the seed entity are considered RIGHT patterns. The output of this transformation step is an RDD containing a list of tuples containing the pattern and the 1 integer.

The output of the FlatMap transformation returns the Tuple containing the Integer instead of just a pattern so that when reducing the results, the Integers are added up, resulting in the count for each of the patterns. The reduction is done through a reduceByKey transformation which shuffles the elements on the RDD so that all the elements with the same keys are on the same machines, and then reduces them.

At this step, a big number of patterns is considered, but most of them are going to be discarded because they appeared before or after a seed entity by coincidence, and do not introduce entities of that certain relation in general.

Company	School	Role	Location
software engineer at _company_	study electrical engineering at _school_	_role_ at company	live in _location_
co-founder at _company_	engineering at _school_	_role_ at ministry	based in _location_
marketing director at _company_	study graphic design at _school_	in charge of _role_	my home in _location_
regional sale manager _company_	master 's degree at _school_	as junior- _role_	company from _location_

Table 4.1: Example of extracted patterns for each relation

4.1.5 Pattern Evaluation

The output of the pattern extraction phase is too big and too diluted. Many patterns were not identified more than once and do not produce valuable output, so an evaluation is needed. In this implementation, the evaluation is done based on two metrics: confidence, and support.

The support of a pattern is the number of times a pattern was extracted and it can be found in the output of the pattern generation (the values for each entry in the output of the previous reduceByKey transformation). Only the support did not prove to be enough, as there are many expressions which are very common in descriptions and headline without introducing entities most of the time (e.g. the patterns "is a" or "as a").

The confidence is based on the report between the number of times a pattern was identified (the support) and the total number of its appearances.

To compute the total number of a patterns appearances proved not to be very straight-forward because the whole list of patterns was needed when iterating through the input. This was not feasible because there were too many patterns to pass between machines (for an input of 172 million profiles, there were 18 million patterns generated). To solve this, an RDD containing the ngrams (size between 2 and 4) from the input was computed. This step takes longer to compute, and takes up a lot of memory, but it contains the total number of times each of the patterns was encountered. By joining the ngram RDD with the pattern RDD, another RDD is created containing both the support and the number of times it appeared. Based on these numbers, the confidence can be computed.

After support and confidence are computed, all the patterns are aggregated from all the nodes and filtered to a set of high confidence patterns using a Spark filter action. All patterns that are above the set thresholds for confidence and support (for each respective relation) are considered confident.

An overview of how Spark operations are used to extract confident patters from the text is provided in the diagram in Fig.4.6.

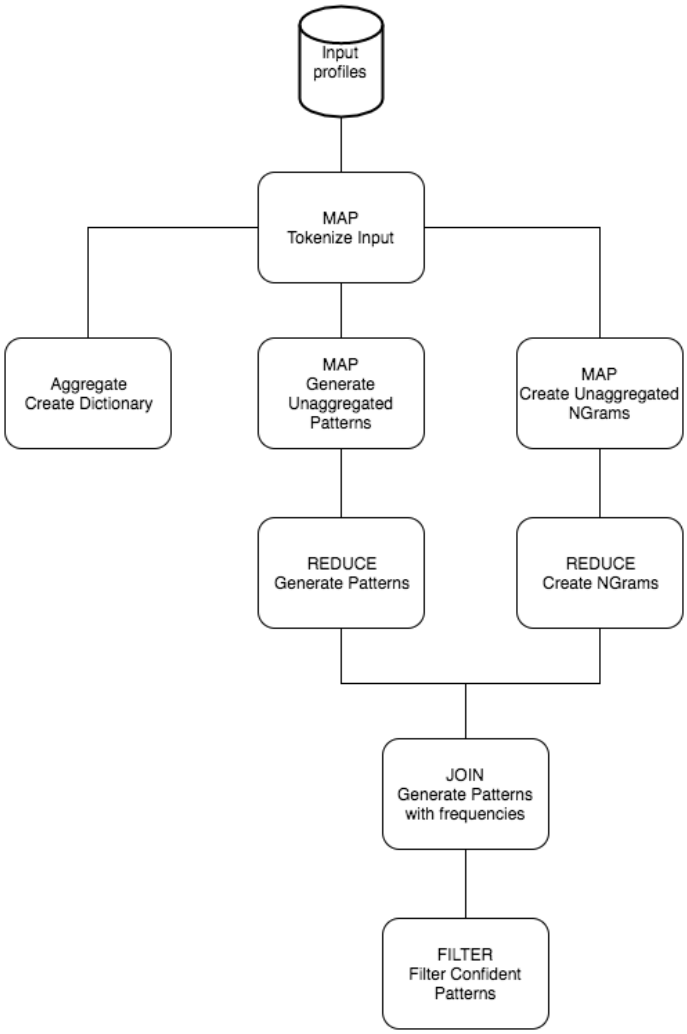

Figure 4.6: Diagram of the pattern generation process

4.1.6 New Entity Extraction

The newly extracted confident patterns are applied to descriptions, headlines and other unstructured texts in order to extract new entities.

The input text of the entity extraction is tokenized in order to have the lemmatized versions of each word. The next step consist of iterating over the list of tokens. Whenever one of the patterns (from the generated hashtable of confident patterns) matches the sequence of tokens, the tokens to the left or to the right (depending on the pattern type) are evaluated as entities. It doesn't always happen that a sequence of words which forms a confident pattern is actually also introducing an entity. Sometimes the word before the RIGHT pattern or after the LEFT pattern can give no information or context related to the relation considered. This calls for a method to evaluate the words in the context of the relation. The factor which seems to be the most relevant is the frequency of the certain word in the seed entities

found when training.

$$RF(w, r) = \frac{wordInRelation(w, r)}{dictionary(w)}$$

where wordInRelation(w, r) is the number of times word w has appeared in an entity from relation r, and dictionary(w) is the number of appearances of word w in the whole training corpus. The two values are computed in the implementation during the training phase and loaded when the extraction of new entities is performed, together with the confident patterns.

There is also the issue of entities formed by multiple words (University of Gothenburg, Abercrombie and Fitch). By only considering the first word, many entities are incomplete and do not provide any extra information (e.g. the "University" entity does not provide enough context). To solve this, Multi Word Entities (MWE) are also considered: after computing the score of the first word, the next words are also considered and the score of the MWE is computed as the average of the scores. If the score of the MWE is bigger than the score of the simple entity formed by the first word, the MWE is kept, and the other is discarded.

```
Result: new Entity
for i=0,input.size do
    for j=MIN_PATTERN_SIZE,MAX_PATTERN_SIZE do
        for k=0,j do
            str += input.get(i+k);
            for Pattern p : patterns do
                if str == p.text then
                    if p.type == LEFT then
                        possibleEntityText = input.get(i+k+1);
                    else
                        possibleEntityText = input.get(i-1);
                    end
                    entity = getEntity(possibleEntityText);
                    MWE = getMWE(input, p);
                    if entity.score > MWE.score then
                        return entity;
                    else
                        return MWE;
                    end
                end
            end
        end
    end
end
```

Algorithm 1: The extraction of new entities

In the case of company, school and location entities, capitalized words are more frequent, so the score can be boosted whenever a capitalized word is evaluated as

an entity. In the case of words that do not appear in the dictionary (there is no way of evaluating them based on the training), the score given is fixed. In the case of companies, the score given is higher than in the case of any of the other relations considered because there is a higher probability of the name of company not showing up at all in the training data.

4.2 Matching Algorithm Integration

The social network people profile matching system takes as input a dataset of profiles and determines which subsets create an identity, or digital persona of a real person. The output of the system is a list of identities, each containing references to profiles from different social networks.

The entity extraction algorithm described in the first parts of this project can be used in order to improve the precision and recall of the matching system by extracting new features from unstructured text. Having more features for some profiles means there is more data the matching algorithm can use when clustering or comparing profiles (profile resolution).

In the training phase of the entity extraction algorithm, patterns are extracted from

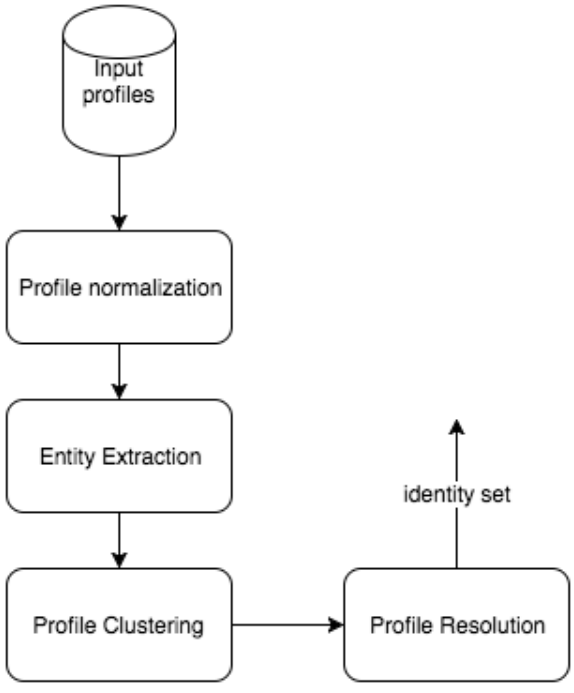

Figure 4.7: Steps of the profile matching algorithm

the training dataset. The patterns, together with the dictionary and relation frequencies are stored in files in an Amazon S3 bucket.

As seen in Fig.4.7, the matching implementation contains multiple stages, which

are distributed over a cluster of machines using the Spark framework. After data normalization, features are extracted from all the structured fields of the profiles. Instead of transitioning to the next step, a static method exposed by the pattern algorithm is called.

The static method for extracting new entities from unstructured text loads all the necessary data only once per workers and outputs the new entities. Based on the type of the entity (the relation it belongs to), the entities are added to the list of entities from structured fields. The extracted entities are then used as features when clustering the profiles and when comparing profiles to determine if they belong to the same person.

5

Evaluation

5.1 Entity Extraction

In this section the entity extraction system is evaluated by first introducing the training and evaluation sets, presenting the effects of varying the algorithm parameters and finally documenting the results in terms of precision and recall.

Training and evaluation are performed on subsets of the whole dataset of profiles. While the dataset contains many profiles that are not very useful as they do not contain much information (work history, education history or location or unstructured field texts), the training and control sets contain mostly profiles from platforms relevant to the task of extracting entities. The training set contains the data from which patterns are extracted (a total of 50 million profiles), while the control set contains semi-labeled entities which can be used to evaluate how well the algorithm works (a total of 30 million profiles). In Fig.5.1 and Fig.5.2 the distribution of profiles is available and in the table 5.1, the number of fields relevant to this task.

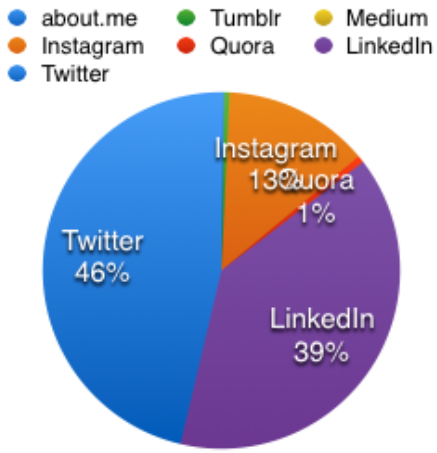

Figure 5.1: Entity Extraction training set

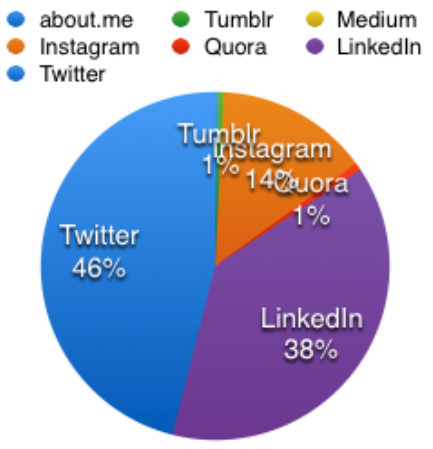

Figure 5.2: Entity Extraction control set

	training set	control set
contains description	10,165,679	6,084,095
contains headline	4,361,079	2,619,221
contains both	4,361,079	2,619,221
contains work history	4,033,181	2,413,189
contains education history	3,492,948	2,031,544
contains location	10,165,234	5,900,609

Table 5.1: Profile fields

The output of the training phase of the algorithm is a set of confident patterns that can be applied to an input text in order to find entities. The control set is not labeled explicitly for this task, however the structured metadata can be considered labeled data. Two main ways of measuring the results of both the entity extraction algorithm and the matching improvements are used: recall and precision. In the context of evaluating the entity extraction system, recall is defined as the percentage of labeled entities which are identified. Precision on the other hand will only be an approximation because there is no way of identifying false positives in the output of the system.

This step is also done in a distributed manner using a Map transformation which takes the control set profiles as input. If the profile contains structured data such as work or education history, the entities are extracted and confirmed if also found in the headline or description. The input is tokenized and the new entity extraction steps are applied to it. After the entities are identified, they are matched with the confirmed entities, which translated to four possibilities: (1) the confirmed entity was not extracted by the algorithm, (2) the exact entity was extracted by the algorithm, (3) the entity extracted by the algorithm contains the confirmed entity and (4) the extracted entity is contained by the confirmed entity. Partial similarity is considered in the case of extracted entities that are bigger or smaller than the actual labeled identity by using the Longest Common Subsequence (LCS) algorithm. This has the advantage of contributing to a clearer recall and precision.

Multiple tests were conducted by varying the training set, the control set and parameters such as the minimum threshold confidence and the minimum support. Also, results are divided by relation, which creates a clearer image of how well the algorithm is doing. The role of the confidence score is to filter out patterns which are very common throughout the purpose, but very seldom appear in the context of a relation, introducing an identity. Their support is high, but their confidence is low. The support of a pattern is used to filter out sequences of words that appear just a few times, not qualifying them to be patterns. Out of the whole generated set of patterns, before filtering, a significant percentage of them has only appeared once or twice.

min_confidence	company	school	location	role	total
0.1	2334	2460	1003	727	6524
0.2	1742	2293	619	332	4986
0.3	1321	2117	411	255	4104
0.4	696	1939	363	135	3133
0.5	357	1784	227	90	2458

Table 5.2: Number of patterns generated depending on the minimum confidence

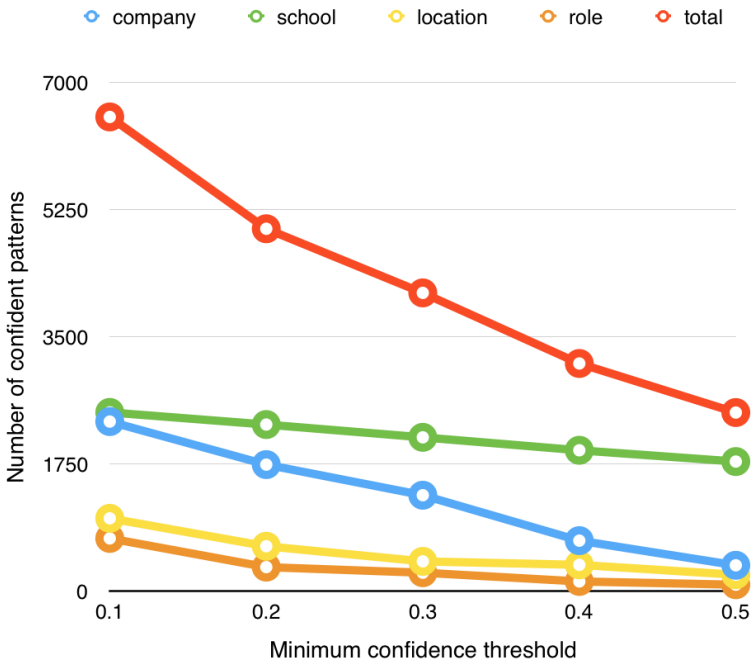

Figure 5.3: Chart of generated patterns depending on minimum confidence

As seen in Table 5.2 and Fig.5.3, the results of varying the minimum confidence used when filtering confident patterns. The same training set, evaluation set and minimum support threshold are used for all three tests. This leads to the same patterns being generated, but as the confidence threshold decreases, the number of confident patterns increases (but not by much). Varying the confidence threshold does not produce important variations in the number of generated results. Lowering the threshold to 0 leads to an explosion of confident patterns, which means that the number of true positives increases, but so does the number of false positives.

The evaluation was performed using the patterns generated in the above discussion (in Table 5.2). The setup with most generated patterns proved to be the best, and the results are detailed in Table 5.3.

	company	school	location	role
exact matches	111,492	49,098	66,322	19,668
extracted is contained by labeled entity	87,004	81,784	11,136	3,164
extracted contains labeled entity	235,876	201,961	76,441	41,580
total extracted	550,518	389,326	280,000	153,033
total confirmed entities	748,243	445,358	498,783	670,325
precision	0.7890	0.8549	0.5496	0.4209
recall	0.7357	0.8742	0.5614	0.2283

Table 5.3: Number of patterns generated depending on the minimum confidence

5.2 Effects of entity extraction on matching

In order to evaluate the effects the entity extraction system has on the matching precision and recall, a control set must be created. Considering the big number of profiles needed, manually matching the profiles is not feasible, so the control set was created based on profiles gathered from the about.me social platform (about.me profiles contain structured data and also links to profiles on other social networks – all provided by the user who creates the profile as illustrated in Fig.5.4).

This control set contains both the about.me profiles and the profiles that are linked, and based on the about.me profiles the precision and recall can be computed.

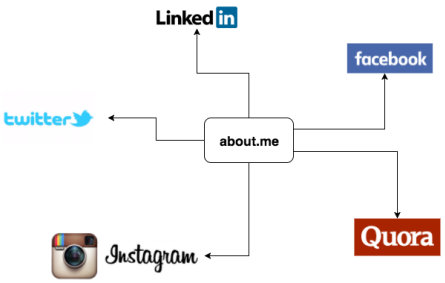

Figure 5.4: about.me relation to other social platforms

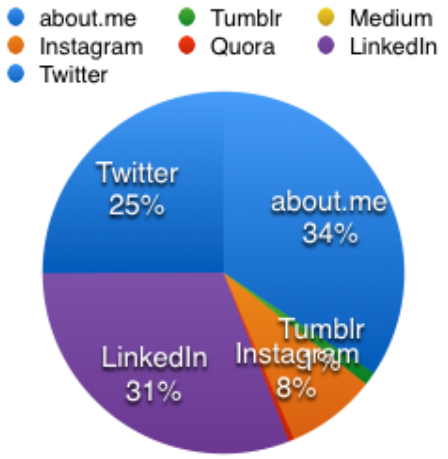

Figure 5.5: The distribution of the profiles in the control set

The control set is stored in a bucket on S3 and is composed of a total number of 355864 profiles with the distribution in Fig.5.5

Out of all the profiles, 101108 of them have some unstructured text in the form of a headline or description, 116429 have information about work history and 107578 have information about education history.

	6524 patterns	4982 patterns	3133 patterns	2458 patterns
school and company entities	63051	48630	10772	6524
location entities	1318	941	110	8
role entities	7260	1617	725	26
precision increase	0.0625	0.0631	0.0672	0.0675
recall increase	0.1274	0.1119	0.0933	0.0919

Table 5.4: Recall and precision improvements using the entity extraction

In table 5.4 the results of plugging the entity extractor into the matching system can be observed. When a small confidence threshold is used, more patterns are extracted which leads to more entities being extracted. When more entities are extracted, there is a bigger chance of extracting false positives, which causes the precision to decrease and the recall to increase.

It can be obvserved that the number of extracted entities decreases a lot when the number of patterns also decreases. This can be caused by some patterns with low confidence that introduce a lot of false entities.

Depending on the goal of the matching system, someone can opt for more recall or more precision. Using more patterns might be more helpful in the sense that the chances of matching badly depending on extracted entities are slimmer.

6

Related Work

The entity extraction task is part of the effort to structure the big amount of data available on the web. A number of papers have been written on this subject, and there are academic (Stanford CoreNLP[10][1]) and commercial (AlchemyAPI[2], Basistech[3] and Lexalitics[4]) software, libraries and APIs available for this task. The research done on this topic seems to revolve around three main methods: the use of statistic models, machine learning techniques (such as neural networks) and pattern recognition algorithms. The focus of this implementation has been the pattern recognition algorithms, as they seem to fit quite well with the data provided as input.

The structure of the bootstrapped algorithm was presented in [7]. As the authors of this paper point out, machine learning techniques for named entity recognition are widely used in academia, but rule-based methods (such as pattern algorithms) are more commonly found in commercial software. This is because running a supervised machine learning algorithm on a big corpus means labeling a big amount of data, which sometimes is not feasible. Another reason is the fact that patterns can outperform the machine learning methods in certain specialized fields, as it is shown in the paper.

Gupta et al. developed their algorithm to work on drug and treatment entities from forums on the Medhelp platform[5] and have used a set of predetermined entities as seeds. Based on the seed entities, the algorithm can be started by looking for the patterns that are created around them. The same approach was chosen in this project, using the structured field of a profile to extract seed entities.

The patterns they used were between 2 and 4 words long, ignored very common words like “a”, “an”, and “the” and were made up of objects containing the lemmatization of the initial words and the part-of-speech tags (generated using the Stanford POS tagger[6]). Lemmatization proved to be very valuable in extracting general

[1] http://stanfordnlp.github.io/CoreNLP/
[2] http://www.alchemyapi.com/
[3] http://www.basistech.com/
[4] https://www.lexalytics.com/
[5] www.medhelp.org
[6] http://nlp.stanford.edu/software/tagger.shtml

patterns, but with a quite big cost in efficiency (lemmatizing so much data is expensive). Also applying a POS tagger proved infeasible with the big amount of data used when training.

A lot of patterns can be generated by considering the window of 2-4 words next to a pre-labeled entity. As a big number of patterns can generate many false-positives, they are further scored based on the number of positive entities they identify, but also based on the number of expect negative entities. In this paper the top k confident patterns are kept, while in our implementation a threshold was used.

The score of an entity belonging to a class C, which is identified by a pattern, and computed based on the average of five features: the edit distance from positive entities, the edit distance from negative entities, the semantic odds ratio (based on the ratio between it's frequency in the positive and negative entities using Laplace smoothing), Google Ngram[7] occurrence statistics (the more common an entity is, the less score it receives) and the distributional similarity score (computed based on how close the identified entity is to other positive entities in a distributional similarity cluster – Stanford CoreNLP Distsim tagger[8]). As the implementation in this project is distributed, getting the entities to the workers proved infeasible, so similarity scores were not computed. Occurrence statistics were the decision factor on keeping an entity.

Using the confident patterns to extract new entities and scoring them based on their frequency and similarity to positive and negative entities, yields a new set of entities that can be used as seeds for a new iteration.

RAPIER (Robust Automate Production of Information Extraction Rules) is documented in [11] and it provides a method of rule extraction based on syntactical information (using a POS tagger) and semantic classes. This is a bottom up (specific to general) approach to finding patterns using an ILP (Inductive Logic Programming) algorithm.

Trained on a corpus composed of newsgroup job postings and a set of pre-filled templates, RAPIER generates rules which are then applied to other job postings. An extraction rule is composed of the pre-filler (optional), the filler, and the post-filler (optional), each containing a list of pattern items (constraints which can be lists of words, POS tags and/or semantic classes). Part of a document matches a rule when at least one in all the constraints is satisfied. As it is a bottom-up approach, the most specific rule is considered, matching all the words in the pattern items, but as this proves too specific, two rules are paired up and the LGG (least general generalization) is computed. Furthermore, if the constraints are disjunct, the rule might prove too specific, so another rule is generated, dropping the constraints altogether.

Because the complexity for generating rules by considering all the words in a corpus is too big, the algorithm iterates starting from the filler, and extending the pre and post fillers. At the point when the best rule (scored based on the length) no longer

[7] http://storage.googleapis.com/books/ngrams/books/datasetsv2.html
[8] http://nlp.stanford.edu/nlp/javadoc/javanlp/edu/stanford/nlp/tagger/maxent/Distsim.html

produces negative examples, the generalization is stopped. This approach generates a small number of very general patterns/rules, which is different from the quantitative approach taken in the pattern algorithm described in this thesis.

When trained on 100 documents paired with their templates, this method was evaluated to a 83.7% precision and a 53.1% recall.

One other relevant task to the extraction of entities is the extraction of relations as it can add more information and context to tuples of entities. The relation extraction task is usually based on fixed predefined patterns (working at, studied at) which can be extracted from text corpora or from knowledge databases such as Freebase[9]. One downside of this is the fact that the equivalences between relations are not used (such as X physicist at Y and X professor at Y).[12]

To improve the number of extracted and predicted relations, the authors of the work introduce asymmetric equivalence between the relations. A matrix is built with rows consisting of tuples of entities, and columns consisting of the relations. The matrix ($\theta_{r,t}$) contains the probabilities that entity e1 in the tuple of entities is in relation with entity e2 from the tuple, where t is the tuple and r is the relation. $\theta_{r,t}$ is computed based on the Latent Feature Model. Furthermore it is interpolated based on other relations for the same tuple using the Neighborhood Model.

In [13] the authors explore the use of Deep Neural Networks (DNN) which are a very popular tool in machine learning at the moment. Daniele Bonadiman et al. describes a DNN model for NER in Italian, which requires few modifications to work for other languages. The method described introduces a way through which, using recurrent feedback, output tag dependencies are considered and also a pre-training technique.

The Context Window Network (CWN, described in Collobert et al. (2011) [14]) takes as input the context (a window of neighboring words) for a sequence s = [w1, w2, .. , wn]. Each words is then mapped to it's word embedding vector (language features) and concatenated into a single vector, which is used by the hidden layer. Next, a linear function M1 x r + b1, M1 being the weights and b1 being the bias, is applied and passed through a HardTanh activation function. The output of the hidden layer is sent to the output layer which applies the softmax function on the M2 x r2 + b2 linear function.

The authors build on the CWN by proposing the Recurrent Context Window Network (RCWN) because of the main drawbacks of the CWN: the fact that there are not dependencies between output tags, and overtraining. In the RCWN the embedding vectors for the output tags [i-m, . . . , i-1], where m is k/2, are used as additional input. The tags j > i are not yet computed, so they are considered as unknown, using the UNK tag.

To prevent overfitting, three methods are combined: weight decay (a regularization term used to decrease the magnitude of the weights), early stopping (a method used

[9]https://www.freebase.com/

for stopping before overfitting) and Dropout (drop units randomly during training to prevent co-adaptation).

The matching of profiles between social networks has been studied in a number of papers. Because collecting the data from multiple social networks is not a trivial task, the problem of matching has been approached in different angles: using a control set of profiles generated with a random word generator or using a small control set of gathered profiles.

In [15], the authors propose a 4 component system which matches a control set of profiles which were generated with a random word generator (part of the profiles generated have a high degree of similarity, simulating the fact that they belong to the same person). The first component is a FOAF (Friend-of-a-friend [16]) Middleware, which converts the different profile types to FOAF profiles (basically a normalization step, which is also done in our implementation). The second component assigns similarity functions to each profile field: (i) Syntactic-based similarity approaches (approximate matching techniques such as Jaro-Winkler distance for names [17]) and (ii) Semantic-based similarity approaches (Explicit Semantic Analysis [18] which uses Wikipedia to compute relatedness). As the fields in a profile vary in importance when matching two profiles, weights are attributed, both manually and automatically. Automatic weight assignment is done by extracting the profiles with the same email address and computing the similarity between their fields. Each field is then assigned a weight computed by averaging or using Fuzzy Decision Trees on the obtained similarities. The final component makes the decision if two profiles are a match by comparing their similarity with a threshold.

7

Ethics

The entity extraction system developed in this thesis is used to improve the recall and precision of a profile matching system. While entity extraction does not raise any ethical questions on its own as it does not directly affect people, a discussion of the ethics of matching people social profiles is in order. The question raised is: "Is it okay to create entities containing all the publicly available profiles of a person, or is it a violation of their privacy ?"

One of the fundamental rights (which has been adopted in over 150 national constitutions) is the right to privacy. This inalienable right has been the focus of a lot of discussions, with an increased attention in the last few years, as a trade-off between privacy and security has been promoted by government agencies (CIA, NSA etc.).

In the case of this project, all the data used is publicly available on the different social platforms, and the task of matching them can be done manually. Doing this automatically can give people a clearer idea of what data is publicly available on them. Considering the average Internet user has 5.54 social profiles but engages with an average of 2.82, it can be speculated that in many cases social profiles can be forgotten by their owners.

Another right that has been debated and put into practice by the European Union and Argentina in 2006 is the "Right to be forgotten" [1]. This right grants individuals more power over their data spread on second party platforms (databases or paper records) by giving them the ability to ask for their information to be deleted. This right is respected by platforms such as Google which takes right-to-be-forgotten requests and delete the data they have on the person requesting it.

Having all the profiles of a person in a centralized manner can be useful in the context of the right to be forgotten as an individual would have a better view of the information publicly available and then would be able to act on it.

One problem that might arise is that some people might not want their profile on one social media platform to be linked to one on another platform. For instance someone having a LinkedIn profile might not want people to also have their Facebook or Twitter accounts where activity is less work-oriented. If the profiles contain

[1] http://ec.europa.eu/justice/data-protection/files/factsheets/factsheet_data_protection_en.pdf

the same name and other specific information such as location, work or education history, they might get matched into an identity which someone can easily access. Considering all the data used for creating identities is publicly available data, and the fact that social networks sites have high quality search functions, the matching system does not have any significant side effects on privacy. The task of manually matching two profiles is not very hard for people when the data is available, which means that a person who wants to track down someone's profiles can easily do it anyway.

8

Conclusion and future work

There are two parts to this project from which conclusions can be drawn: the entity extraction algorithm and the effects it has on the matching accuracy.

The algorithm used for entity extraction was chosen as to benefit from the structure of the dataset of social network profiles. In order to extract as many accurate patterns as possible, the data was distributed on a cluster of computers using the MapReduce paradigm. This allowed the processing of a lot of data, but it had its own drawback as it limited the evaluation methods. In a few cases during the implementation a few workarounds were needed (such as using ngrams in order to compute the number of times a pattern has appeared in the whole corpus). The workarounds were usually computationally expensive, so part of the benefit of distributing the data was lost. Another important fact is that most entities are extracted using the most frequently used patterns which can be identified with a relatively small training set. Extracting patterns from a big amount of data yielded small improvements in the number of correct entities extracted.

Another aspect which can be considered important is the fact that when working with millions of profiles, the most obvious solutions proved to have the most impact. Extracting co-occurrence between words or conjunctions added a lot of complexity, but did not improve the results by too much.

The fact that patterns were extracted from profiles is helpful as the entity extractor is to be used in the context of profile matching, but the algorithm is general enough that it can be applied to any corpus.

Even though the entity extractor seems to have good results, the impact it has on matching is not very big. Intuitively when starting this project we believed that extracting information from the unstructured fields of a profile would significantly improve matching recall and precision. Evaluating the impact shows that even though a big number of entities is extracted, mostly with companies and schools, precision and recall have a limited increase. A closer analysis of the data shows that most profiles which contain unstructured text also contain detailed information which would have been matched anyway, and the number of profiles that benefit from the entity extraction is relatively small.

As future work, the new entity extraction step of the system described in this thesis can be developed further by implementing statistical methods. The generated patterns are used to give an indication that a word or collection of words might be an entity, but more methods can be used to determine that with a degree of certainty. Also part-of-speech tags can be added using the Stanford POS tagger[1], which might prove useful in generating good patterns and extracting new entities. Another analysis can then be done on the effect entity extraction has on matching.

[1] http://nlp.stanford.edu/software/tagger.shtml

Bibliography

[1] Lisa F Rau. Extracting company names from text. In *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, volume 1, pages 29–32. IEEE, 1991.

[2] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2009.

[3] David A Smith. Detecting and browsing events in unstructured text. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 73–80. ACM, 2002.

[4] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

[5] Maeve Duggan, Nicole B. Ellison, Cliff Lampe, Amanda Lenhart and Mary Madden. Frequency of Social Media Use, January 2015. `"http://www.pewinternet.org/2015/01/09/frequency-of-social-media-use-2/"`, Accessed: 2016-05-15.

[6] Jason Mander. Internet users have average of 5.54 social media accounts, 2015. `http://www.globalwebindex.net/blog/internet-users-have-average-of-5-social-media-accounts`, Accessed: 2016-05-15.

[7] Sonal Gupta and Christopher D Manning. Improved Pattern Learning for Bootstrapped Entity Extraction. In *CoNLL*, pages 98–108, 2014.

[8] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. Spark: Cluster computing with working sets. In *2nd USENIX Workshop on Hot Topics in Cloud Computing, HotCloud'10, Boston, MA, USA, June 22, 2010*, 2010.

[9] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[10] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[11] R Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 328–334, 1999.

[12] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 74–84, 2013.

[13] Daniele Bonadiman, Aliaksei Severyn, and Alessandro Moschitti. Deep Neural Networks for Named Entity Recognition in Italian. In *Proceedings of the Second Italian Conference on Computational Linguistics CLiC-it 2015*, page 51. Accademia University Press, 2015.

[14] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[15] Elie Raad, Richard Chbeir, and Albert Dipanda. User profile matching in social networks. In *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, pages 297–304. IEEE, 2010.

[16] Li Ding, Lina Zhou, Tim Finin, and Anupam Joshi. How the semantic web is being used: An analysis of FOAF documents. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 113c–113c. IEEE, 2005.

[17] William W Cohen, Pradeep D Ravikumar, Stephen E Fienberg, et al. A Comparison of String Distance Metrics for Name-Matching Tasks. In *IIWeb*, volume 2003, pages 73–78, 2003.

[18] Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *IJCAI*, volume 7, pages 1606–1611, 2007.

A

Appendix 1


```
{
    "profileType": "linkedin",
    "username": "",
    "url": "",
    "name": "",
    "headline": "AC/DC Supervisor at Electrix",
    "description": "",
    "positions":[
        {
            "company":"Electrix",
            "role":"AC/DC Supervisor"
        },
        {
            "company":"PBA Ltd",
            "role":"Technical specialist"
        }
    ],
    "companies":[
        "PBA Ltd",
        "Transfield Services"
    ],
    "education":[
        {
            "school":"Waitaki boys high"
        }
    ],
    "locations":[
        {
            "raw":"Canterbury & West Coast, New Zealand"
        }
    ]
}
```