CA' FOSCARI UNIVERSITY – VENICE
Department of Environmental Sciences, Informatics and Statistics
MSc in Computer Science

Master thesis

Candidate: Francesco Pizzolon

SEED: A Framework for Extracting Social Events from Press Reviews

Examiner: Prof. Salvatore Orlando

Co-examiner: Prof. Flavio Sartoretto

Academic year 2011-2012

A Giuseppe, Irene, Michele.
A Mario e Angela.
A Primo e Ines.

Abstract

In the last two decades, a huge amount of data are increasingly become available due to the exponential growth of the World Wide Web. Mostly, such data consist of unstructured or semi-structured texts, which often contain references to structured information (e.g., person names, contact records, etc.). *Information Extraction* (IE) is the discipline aiming at generally discover structured information from unstructured or semi-structured text corpora.
More precisely, in this report we focus on two IE-related tasks, namely *Named-Entity Recognition* (NER) and *Relation Extraction* (RE). Solutions to these are successfully applied to several domains. As an example, Web search engines have recently started rendering structured answers on their retrieved result pages yet leveraging almost unstructured Web documents.
Concretely, we propose a novel method to infer relations among entities, which has been tested and evaluated on a real-world application scenario: entertainment event news, where starting from a generic press review, we try to discover new events hidden in it. Our method is subdivided in two steps, each one specifically addressing an IE task: the first step concerns NER and uses a knowledge-based technique to correctly and automatically identify named entities from unstructured text news; the second step, instead, deals with the RE task, and introduces a novel, unsupervised learning strategy to automatically infer relations between entities, as detected during the first step.
Finally, well-known measures over a real dataset have been used to evaluate the two parts of the system. Concerning the first part, results highlight the quality of our NER approach, which indeed performs consistently with other existing, state-of-the-art solutions. Regarding the RE approach, experimental results indicate that if enough relevance can be found on the Web (in our case, documents concerning the candidate event), it's possible to infer correct relations which lead to the discovery of new events.

Contents

1 Introduction **7**

2 Background and Related work **10**
2.1 Information Extraction 10
2.1.1 Applications 13
2.2 Named Entity Recognition 17
2.2.1 Knowledge based methods 18
2.2.2 Rule based methods 19
2.2.3 Statistical methods 24
2.2.4 Evaluation 28
2.3 Relation Extraction 30
2.3.1 Supervised methods 31
2.3.2 Semi-supervised methods 36
2.3.3 N-ary relations 42
2.3.4 Evaluation 44

3 Real world scenario **45**
3.1 Company profile 45
3.2 Editorial office and press reviews 46
3.3 Definition of the problem 48

4 Approaches and implementation **51**
4.1 Strategy 51
4.1.1 SEED - Social Entertainment Events Detection 54
4.1.2 Implementing language 55
4.2 NER - Named Entity Recognizer 56
4.2.1 Date tagger 57
4.2.2 Range tagger 60
4.2.3 Sentence splitter 61
4.2.4 Location tagger 63
4.2.5 Place tagger 66

4.2.6 Artist tagger 69
4.3 RE - Relation Extractor 73
4.3.1 Candidate Extraction 73
4.3.2 Candidate Ranking 77

5 Experimental results **81**
5.1 NER evaluation 81
5.2 RE evaluation 82

6 Conclusions **85**

7 Future works **87**

List of Figures

2.1 Example of entity extraction 11
2.2 *Tweets* from a politician leader 12
2.3 Snapshot of *Google News* 13
2.4 First search result for "pagerank" in *Google Scholar* 14
2.5 Result of a search for a product in *Kelkoo* 15
2.6 A tiny grammar modeling paper citations 25
2.7 Results of a classification problem 29
2.8 A parse tree with positive and negative samples 34
2.9 Architecture of Snowball 39
2.10 Steps for training TextRunner's self-supervised learner 41
2.11 A graph constructed to detect a 3-ary relation 43

3.1 *2night*'s information flow 46

4.1 A *tweet* about our sample event 52
4.2 A blog title about our sample event 53
4.3 A facebook post about our sample event 53
4.4 SEED architecture 54
4.5 Date tagger module 57
4.6 Applying REs with longest match criteria 59
4.7 Range tagger module 60
4.8 Sentence splitter module 61
4.9 Location tagger module 63
4.10 Two famous italian municipalities 64
4.11 Place tagger module 66
4.12 Some locals affiliate with 2night 67
4.13 Artist tagger module 69
4.14 Fragment from Anna Calvi's italian Wikipedia page 70
4.15 Candidate Extraction module 74
4.16 Candidate Ranking module 77

List of Tables

2.1 Tokenization of a sentence 24
2.2 Segmentation of a sentence 25

5.1 NER evaluation results 82
5.2 RE evaluation results 84

Acknowledgments

I am grateful to prof. Salvatore Orlando and Gabriele Tolomei for their essential contribution and for their continuous support; without their help this work would not have been of the same scientific quality.
A thank also to Daniele Vian at 2night for providing the material used in the implementation of the system and for my experimental results.

Chapter 1

Introduction

This thesis aims to accurately describe our solutions for a problem proposed by a company affiliate with our university: detecting new events from press reviews. The company's mission is to advertise both locals present in various italian districts and their entertainment events, which involve the performance of many national and international artists. In this scenario, events are hand recognized from journalists of the company's editorial office by reading and analyzing verbose, long and ambiguous press reviews; this process is often prolix and lead to a waste of working hours, a challenging task is to automate it.
This problem concerns *Information Extraction* (IE), a discipline aiming to extract structured information from unstructured sources of various nature. More precisely, two IE-related subtasks are considered:

1. *Named Entity Recognition* (NER), which expects to extract and classify entities from unstructured text. In our scenario this translates into detecting entities of the classes *Date*, *Range* (entities formed by continuous dates), *Location* (municipalities in Italy), *Place* (locals affiliate with the company) and *Artist* from press reviews.

2. *Relation Extraction* (RE), aiming to extract semantic relation between entities. In our case, relations are represented by 3-ary tuples connecting our entity classes in this way: $(Date, Location, Artist)$, $(Range, Location, Artist)$, $(Date, Place, Artist)$ or $(Range, Place, Artist)$; these tuples model entertainment events indicating that an artist is performing in a certain place or location on a precise date or set of dates.

After analyzing some state of the art solution for both the tasks cited, we decided to define a novel strategy for RE: exploiting the potential of the Social Web to infer our relations. Actually, well known solutions work by

sentence level and regard only the single document to extract relations but, in our scenario, relations can span over the single sentence and even cross documents. For example, it's not always true that if an artist, a place and a date are named in the same sentence they represent the right entertainment event, reflecting press reviews verbiage.
Observed this, we decide to implement a framework called *Social Entertainment Event Detection* (SEED) based on our new approach. SEED works together with an external module, called "Fresh Social Knowledge" (FSK) which allows to rank relations between entities by providing other documents about the candidate events. Documents retrieved by the FSK are analyzed by SEED which scores candidate tuples; in the last phase tuples corresponding to potential new entertainment events (i.e. with a score much higher than other candidates) are returned.
Finally, well known measures over a sample set of press reviews will be used to evaluate our framework together with two baselines used for comparison, and conclusions will be outlined regarding experimental results obtained.
A brief introduction of every chapter is presented, so the user can skip to the chapter he's interested in:

- First chapter represents the introduction, it introduces the problem threated and it explains how the dissertation is structured by giving a trace of what is contained in every chapter.

- Second chapter contains the background one must know to understand our work: the first section is dedicated to the IE task and its applications, while the second and third subsections explain state of the art solutions used for NER and RE tasks according to surveys [1] and [2].

- Third chapter describes the real world scenario where we employ our new method; it contains two sections about the company profile and its editorial office together with a third section which defines formally our problem. In this chapter a generic press review is reported and it will be recalled as example when SEED's submodules will be explained in the following chapter.

- Fourth chapter defines motivations and the strategy adopted to resolve our problem: each submodule of the framework SEED is documented in details and its use is shown using the press review of example.

- Fifth chapter outlines how the system has been evaluated and reports evaluation's results of our framework and two baselines used of comparison. Finally, results are compared and explained.

- Sixth chapter summarizes the dissertation and reports conclusions that experimental results highlight.
- Seventh chapter is dedicated to the possible future works and directions inspired by topics addressed by this work.

Chapter 2

Background and Related work

This chapter is dedicated to the background knowledge one must have in order to understand our work. In the first subsection, the task of IE is generally described also bringing some example of its applications in the real world; the second and third subsection are about the NER and RE tasks, giving formal definitions of the two problems and facing principal state-of-the-art methods to address them.

2.1 Information Extraction

The task of automatically extract structured information from unstructured or semi-structured sources is called *Information Extraction* (IE). This challenging task, that still engages the attention of many researchers, has opened new scenarios to organize, analyze and query data, assisting *Information Retrieval* (IR) in his role.
The types of structure extracted from unstructured sources can be subdivided into four categories:

- Entities: they are noun text snippets like name of companies, people, places, locations;
- Relationships: they are defined over two or more entities and they outline a certain correlation between them. A typical relation between a company and a person is "is-CEO-of", meaning that a person is the Chief Executive Officer of the company;
- Entity's adjectives: generally they are words which describes an entity, for example, if a *tweet* speaks about a politician, we are interesting in how the critique to that public person can be classified;

- Lists, Tables and Ontologies: if we desire to have a strong relation over data, we can store extracted information in those types of structure, e.g. if we search for all the books published by a certain author and a table which fields are author and books is available, we can just query this table selecting the author and present the results.

Figure 2.1: Example of entity extraction

Figure 2.1 shows a typical example of entity extraction from a news of a famous space agency: companies are highlighted in blue, places in orange and people in green.
The unstructured sources used for the extraction can be distinguished by their granularity and by their heterogeneity in terms of documents that compose them, and they can be subdivided in:

- Sentences or Records: they represent the most common unstructured source, and they are small text snippets. Examples of this kind of source are *tweets* from *Twitter*[1], personal status from *Facebook*[2], classified ads or sentences extracted from a paragraph of a book. Figure 2.2 shows an example of this type of unstructured source.

- Paragraphs and Documents: some tasks involve the analysis of a whole paragraph or an entire document to correctly perform an extraction. Examples can be searching for the causes and the consequences of a disease in a medical report, extract the subject of a talk in a conference, find new events in online news and extract user technical issues from emails in a call center.

[1]http://twitter.com
[2]http://www.facebook.com

- Machine generated Pages: they are documents generated from a machine, and therefore they can contain a predefined structure; an example is a web page generated with HTML5 that contains an article: one can easily access it because of its particular HTML tag.
- Partially structured Domains: this category contains sources with a well defined scope and domain, say news articles, videogame blogs or movies review sites.
- Open Ended Sources: sometimes there is interest in extracting knowledge in environments like the whole Web, which has a lot of heterogeneity and it's not consistent.

Figure 2.2: *Tweets* from a politician leader

The concept of IE comes from a competition in the late 80s, the *Message Understanding Conference* (MUC) which, with seven editions, has been based on different domains like fleet operations [3], terrorist activities in Latin America [4], corporate joint ventures and microelectronic production [5], negotiation of labur disputes and corporate management succession [6] and airplane crashes and satellite launches [7]. The importance given from MUC quickly led to the development of IE systems, which were successively tuned and tested by participant teams during competitions.

2.1.1 Applications

IE is useful in various contexts and we list here, with some examples, four mayor categories of applications that require or are totally based on this task.

Enterprise Applications

News tracking. A very popular application of IE is to extract news and facts from various news sources. For example, the MUC and another competition, the *Automatic Content Extraction* (ACE), are based on the extraction of entities and relations between them in different domains. A recent example is *Google News*[3], which aggregates news from a vast set of sites (the English version counts more than 4500 web sources) to give an overview of what's happening in a specific country in the last thirty days. An example of this application can be seen in Figure 2.3 that shows five important news reported by main online newspapers in the United States on the 25th January 2013.

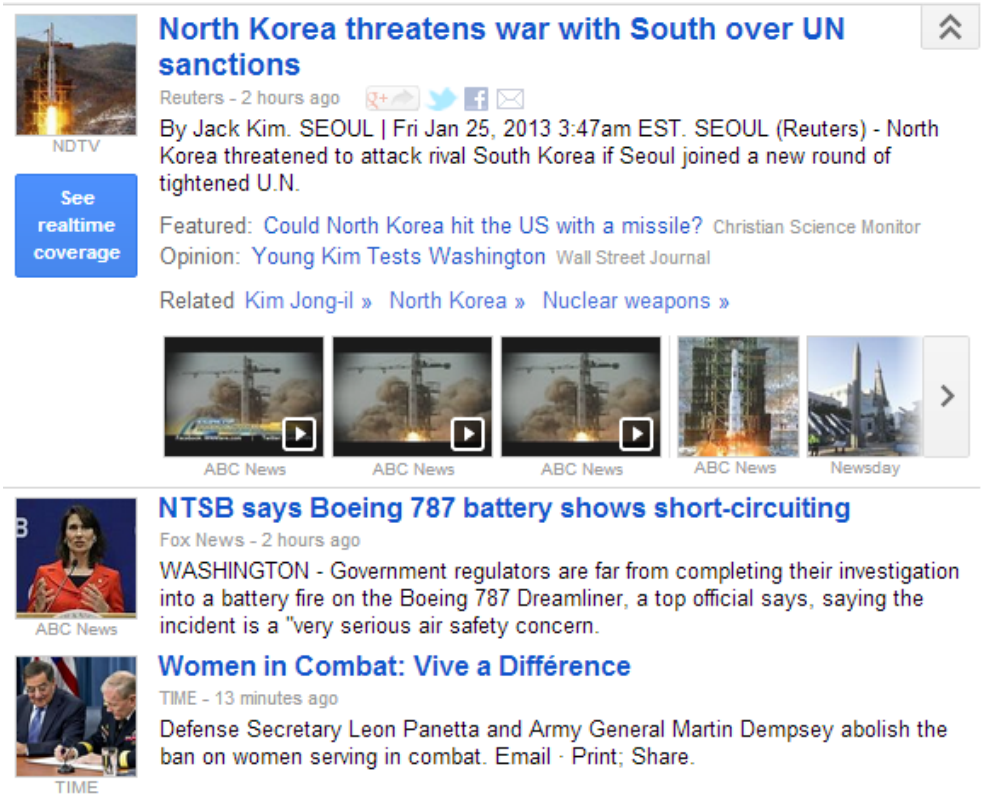

Figure 2.3: Snapshot of *Google News*

[3]http://news.google.com

Data cleaning. This is one of the basic task to maintain a database clean; for example, a lot of organization such as governative organization, banks and transport companies store a lot of addresses of their customers, and they're potentially interested in subdivide the address in fields in such a way that it's automatically understandable the road, the city and the country the customer lives.

Web Oriented Applications

Citation databases. An example of this application is *Google Scholar*[4] : launched in 2004, it allows the research of academic literature, and for this reason it requires to investigate various unstructured sources like papers, publications, public talks and PDF documents extracting authors, titles, dates, references and other features that each academic research contains. The result is a structured database and it's possible to query it, for example searching a paper by its title or its authors; it is also possible to know some statistics about a paper, like how many other researches cite that paper. Figure 2.4 shows the first search result for the world "pagerank" in *Google Scholar*, showing that the famous paper of S. Brin and L. Page is currently cited by other 5489 researches.

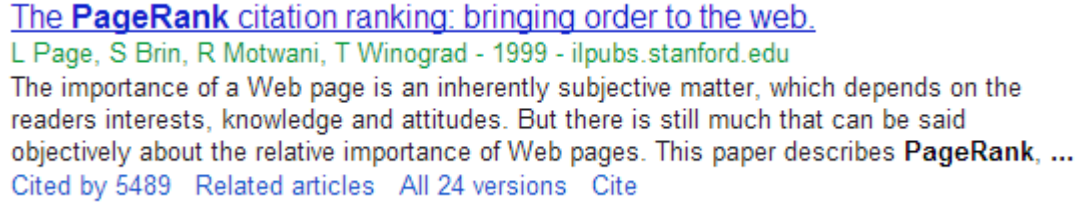

Figure 2.4: First search result for "pagerank" in *Google Scholar*

Comparison shopping. Nowadays, people often use the Web and its services to buy products, and there exist services that compare the products by crawling online stores; the information extracted allow the user to list the sites that offer the minor price for the product he's interested in. An example is *Kelkoo*[5], a comparison shopping web site, operating in ten European countries. Figure 2.5 shows various prices of a famous camera in different online stores.

[4]http://scholar.google.com
[5]http://www.kelkoo.com

Figure 2.5: Result of a search for a product in *Kelkoo*

Scientific Applications

Disease intelligence: a recent scientific application of IE is represented by finding information about diseases by scratching data from various unstructured sources: disease peculiarities change from a part of the world to another, and it's important to share those information between researchers and doctors.

From the examples reported above, it's clear that nowadays IE is become essential, just think about the vastness of the Web: only a small part of it can be defined as structured and recently, with the huge diffusion of smartphones and handheld devices together with data we generate using them every day, unstructured information has grow up exponentially. Also, an hypothesis widely accepted is that the 80% of data usable in business processes come from unstructured sources; last example is that in January 2013, *tweets* became a measure of audience about television shows, giving a fundamental role in identifying active audience of television programs and conditioning choices of advertisers.
IE plays also an essential role in today's web search engines; indeed, before

using classical techniques of IR to give results, they preprocess the query with IE techniques, for example detecting entities hidden inside the search keywords and using this knowledge to give better search results.

2.2 Named Entity Recognition

In this section we generalize the NER problem and we describe three main approaches to address it: knowledge based methods, rule based methods and statistical methods. First subsection is dedicated to the naive approach that involves the use of gazetteers. Second subsection describes rule based methods: we list rule forms and give an example of two well-known algorithms for learning rules from labeled examples. Third subsection is about statistical methods, where we describe three state-of-the-art models and how them can be used.

Named Entity Recognition (NER) is the task of extracting entities from unstructured text. Figure 2.1 shows an example of this: starting from an article of a famous space agency, we want to extract ad classify entities hidden in it; words colored in blue represents the companies, words in orange represent the places and words colored in green represent people. To understand better the NER task, we state here its formal definition.

NER PROBLEM: Let D denote a generic text document containing n entities, $e_i, 0 < i \leq n$ denotes an entity; let $C = \{c_1, c_2, ..., c_m\}$ denotes a set of m entity classes, $c_j, 0 < j \leq m$ denotes an entity class; we want to extract all the entities cited in D and classify them in a class contained in C; we use this notation (e_i, c_j) to indicate the referent class c_j of the entity e_i.

As an example, consider the problem specified above in these terms:

D = "**NASA**, the **Russian Federal Space Agency** and their international partners have selected two veteran spacefarers for a one-year mission aboard the **International Space Station** in 2015. This mission will include collecting scientific data important to future human exploration of our solar system. **NASA** has selected **Scott Kelly** and **Roscosmos** has chosen **Mikhail Kornienkov**."
$C = \{Company, Person, Place\}$

In this example, we have three classes of entities and we want to find all instances of those classes in D. A named entity recognition system with this input should return as classified entities the set:

E = { (NASA, *Company*), (Russian Federal Space Agency, *Company*),

(International space station, *Place*), (Scott Kelly, *Person*), (Roscosmos, *Company*), (Mikhail Kornienkov, *Person*) }.

We present now three different main strategies to the NER problem stated above.

2.2.1 Knowledge based methods

A naive technique to recognize named entities is using *gazetteers*, i.e. dictionaries or lists of entities. This seems reasonable, because if we want to find an entity we know, we just add it to the dictionary and it will be recognized by a search over it. Knowledge based approaches work in two phases:

1. Large dictionaries of entities are created consulting various sources, each class we are interested in must have its own dictionary in order to give a class to the entities recognized.

2. Simple heuristics, like exact match or similarity score, are used to identify and classify entities in the unstructured documents.

This method offers often high precision and it's fast, because one simple search in the dictionary will lead to the discovery of a new entity. We can also say that the accuracy that this approach give relies on the size of dictionaries used. However, we need to take into account also drawbacks:

- Dictionaries needs to be continuously updated, think about if we were interested in the class *Student*. This is a very general class always in evolution: people often apply for new courses of various genre, and at the same time many students leave their courses; this means that the correspondent dictionary must be always updated.

- Ambiguity resolution is difficult: if two entities belong to two different dictionaries, resolving this ambiguity involve difficult other procedures.

- For new type of entities, dictionaries are often manually created. This requires quite an effort, it is error-prone and subjective.

On the other hand, it has been demonstrated in [8] that small dictionaries of well-know entities are sufficient to obtain valuable results in NER. In general, knowledge based methods are very useful when the entities to be detected can't be identified by any predefined rule and it does not exists any labeled example of the entities, as well as when the interested domain is closed.

2.2.2 Rule based methods

All initial systems designed for NER supported the rule based approach instead of the statistical approach: the reason is that this latter approach needs large corpus of labeled data to train a classifier, and it was almost hard to find this resource at that time. Rule based methods perform consistently when the task is under a controlled domain, for example, it is easy to understand the house number, the building, the road, the city, the state and the zip code from an US address, because it has some predefined formatted forms. A typical system based on this approach is subdivided in two phases: first phase aims to find the entities starting from a collection of rules properly designed by a language expert, while second phase aims to control and correct the output of the first phase with some policies.
A generic rule is usually in the form $\{Pattern \rightarrow Action\}$, where the pattern is a set containing one or more labeled patterns capturing properties of the token which represents the entity and the context of this token. After defining the general form of a rule, we need to describe what's a token and list some common features of it.
A token is a text segment, and one can associate to it the following features:

- String: the string composed by one on more word representing it;
- Part of Speech (POS): syntactical meaning (e.g. noun, verb, etc);
- Frequency: how many times it appears in a collection;
- Orthography: how the words composing the token are written (e.g. capitalized, lower case, etc).

In the following paragraphs we describe principal forms of rules, how to organize them and two algorithms to learn rules from labeled examples.

Single entity identification

To identify a single entity we can use rules composed by three different patterns: a pattern that matches the token which represents the entity and two optional patterns modeling the context before and after the entity. As an example, we present three different rules that can be respectively utilized to extract people, companies and cities from text documents written in English. First example rule is:

$$(\{Sir\}\{Ortography = capitalized\}[2] \rightarrow Person),$$

composed by a pattern matching the context before the entity and a pattern matching the token which represents the entity; the rule will mark as a person all strings which are composed by two capitalized words which have the word *Sir* before them (e.g. Sir Alex Ferguson).
Second example rule is:

$$(\ \{Ortography = capitalized\}[1]\{Inc\} \rightarrow Company),$$

composed by a pattern matching the entity and a pattern matching the context after the entity; this rule will mark as a company all text segments composed by one capitalized word having the word *Inc* after it (e.g. *Google* Inc).
Last example it's the rule:

$$(\{New\}\{Ortography = capitalized\}[1] \rightarrow City),$$

this rule's type it's the same than the first rule, but this rule will mark as a city all text segments which are composed by one capitalized word having the word New before them (e.g. New York).

Multiple entities identification

Some entities are easy to identify when they are together in a form of a regular expression, and in this case we need a specific rule describing how to match more than one entity at a time. For example, suppose we have some text comments describing the rank of an hotel respect to others and a rule like:

$$\{\text{classified}\}\ (\{\text{Orthography} = \text{Digit}\}):\text{Rank}\ \{\text{out of}\}\ (\{\text{Orthography} = \text{Digit}\}):\text{Sum} \rightarrow \text{Ranking} = :\text{Rank, Total} =:\text{Sum},$$

this rule will identify two entities at the same time: the rank and the total number of hotels for each comment.

Mark entity boundaries

The NER task can be seen also as inserting standard predefined tags to a text document in order to delimit some phrases which form the entities. This point of view is very useful in some contexts, especially when it's more efficient to understand where a certain entity starts and where it finishes instead to try to define a specific rule for that class of entities. Typical examples are addresses, that can be composed by six or more different fields, and paper names, which are usually formed by many words. An example is the rule:

$$(\{\text{Ortography} = \text{number}\}\text{:start } \{\text{Ortography} = \text{capitalized}\}[6\text{-}7]$$
$$\{\text{Ortography} = \text{zip}\}\text{:finish} \rightarrow \text{insert} < address > \text{before:start, insert}$$
$$< /address > \text{after:finish}),$$

which will process correctly the address written in this document:

Bob change his residence to 323 Golden Gate Avenue, San Francisco, California, 94102.

marking it as:

Bob change his residence to $< address >$ 323 Golden Gate Avenue, San Francisco, California, 94102$< /address >$.

Organizing rules

A typical rule based NER system is formed by several rules, each of them can be applied for detecting one or more entities. Each rule has a peculiar span generated by text segments in which the rule works; therefore, we can compare the rules to regular expressions and the way we apply them to grammars. Like grammars, in rule based approach conflicts in applying rules may rise when their spans overlaps, i.e. when a text segment, or part of it, is matched by two or more rules. To resolve this problem, we list three basic approaches:

- Custom policies. In this approach, rules are considered having disjunct spans and treated like an unordered collection; they are directly applied to the unstructured document and, when a conflict between two different rules occurs, policies are used to resolve it. For example, a valid policy can be the longest match: if a rule cover more text span than another rule, the first rule is selected.

- Ordered set. Rules are considered as an ordered set, and the order relationship is given by a priority score previously assigned to each rule: if a conflict occurs, the rule with higher score is selected. Typically, the priority score assigned to each rule is derived from the rule's precision calculated over a training set of data.

- Finite state machines. Here, entities are represented as nodes of a *Finite State Transducer* (FST), while rules are represented by the edges that connect the nodes. Each edge condition must be satisfied when walking around the nodes. If the FST is previously defined, there are

no conflicts between decisions in applying rules because each rule is codified by a different path. However, this approach needs also hand coded policies in case a node is connected by more than one edge.

Rule learning algorithms

We stated above that a rule based NER system is formed by a set of hand coded rules and policies to apply them. However, if sufficient labeled example of a class of entities are available, it's possible to automatically infer rules with learning algorithms. The main goal of these algorithms is to identify the smallest set of rules which cover a large part of the training labeled examples; a general framework used to automatically learn rules from a generic document D containing labeled entities it's the following:

Learning rules framework

```
1 RuleSet = ∅;
2 while (∃ entity e ∈ D not covered by RuleSet):
3   Generate new rules around e;
4   Add generated rules to RuleSet;
5 Remove redundant rules from RuleSet;
6 return RuleSet
```

Moreover, algorithms are subdivided in two categories: (1) bottom-up, where specific rules are generalized, and (2) top-down, where a generic rule is specialized.
An example of the bottom-up approach is an algorithm described in [9] called *Learning Pattern via Language Processing* $(\text{LP})^2$, where rule actions are insertions of starting and ending tags, as in the marking entity boundaries fashion. When $(\text{LP})^2$ induce rules, labeled examples that contain the rule action are positive instances, while other examples are negative. Here's its pseudocode:

$(\text{LP})^2$ bottom-up

```
1 RuleSet = ∅;
2 ∀ tag T:
3   Create a seed rule from an uncovered instance of T;
4   Generalize the seed rule and add it to RuleSet;
5   Remove instances covered by new rule;
6 until all positive instances are covered;
7 return RuleSet
```

To create a seed rule it's sufficient to pick the w tokens to the left and right of the entity and associate the tag T; from this operation, a very specific rule comes out and has the form:

$$x_{i-w}, ..., x_{i-1}, e, x_{i+1}, ..., x_{i+w} \rightarrow T,$$

where e is the entity labeled as T and x it's a generic token.
To generalize a seed rule, tokens picked in the previous step are generalized by a feature of themselves; for example, if all previous token are capitalize words, they can be substituted by the rule {Orthography = capitalized}.
$(LP)^2$ has also a top-down approach, more efficient than the one described above. The new version of the algorithm starts from a very generic rule which covers all the positive instances, but at the same time has low precision, and try to specify this initial rule in various ways to reach more precision. Let R_1 be the initial rule consisting in defining conditions at a distance of $2w$ from the entity, like the example in the bottom-up approach. The method starts from the rules which are formed by only one of the $2w$ conditions and forms a set in which the coverage of each rule is at least a predefined threshold s. The algorithm can be specified by this pseudocode:

$(LP)^2$ top-down

```
1 R_1 = rules imposing a condition over one of the 2w positions;
2 for L = 2 to 2w:
3   R_l = set of rules formed by intersection two rules from R_{l-1}
4         that agree on L-2 conditions and differ on only one;
5   Cut rules from R_l with coverage less than s;
6 return R_l
```

For a specific domain, usually, no labeled examples can be found, and rules are hand specified by a language engineer. However, if enough training corpus are available, using rule learning algorithms it's faster and lead to better results than coding rules by hand.

2.2.3 Statistical methods

Statistical methods convert the NER task to a problem of decomposing the unstructured input document in text fragments and properly labeling them in order to form entities. In the following paragraphs we describe main decomposition models, which differ one from each other by their granularity working level, and some training and inferencing methods to use them.

Token-level models

This approach represents the most common form of decomposition: the unstructured document is viewed as a sequence of words separated by punctuation (e.g. dots, commas, quotes) forming tokens, and the scope consists in correctly labeling these tokens; therefore, the output is represented by a sequence of labeled tokens. However, as we have seen before, entities are often formed by more tokens, and for this reason it's a common practice to adopt a popular labeling system known as BCEO (e.g. B=Begin, C=Continue, E=End, O=Other) which decomposes each entity label as where it begins, where it continue and where it ends. Here's an example of the output of an algorithm, based on a token model, searching for people entities,

i	1	2	3	4	5	6	7	8
x	B.	Rodgers	slams	unacceptable	L.	Suarez	diving	comments
y	*B*	*E*	*O*	*O*	*B*	*E*	*O*	*O*

Table 2.1: Tokenization of a sentence

where $x = x_1, x_2, ..., x_i, ..., x_n$ denotes a sequence of tokens and $y = y_1, y_2, ..., y_i, ..., y_n$ the correspondent sequence of labels. Typically, to classify a token, its features and its neighbors features are used (e.g words, orthography, presence in a given dictionary) with a logical classifier or a *Support Vector Machine* (SVM).

Segment-level models

In a segment model, the input document is subdivided into chunks, i.e. text segments composed by several words. A common method to parse chunks is via natural language parsing techniques. Instead of assigning a label to each word, the label is assigned to each chunk that compose the whole entity. Here's an example, with the same input document as token-level models:

i	1	2	3	4	5	6	7	8
x	B. Rodgers		slams	unacceptable	L. Suarez		diving	comments
s	Person		*Other*	*Other*	Person		*Other*	*Other*

Table 2.2: Segmentation of a sentence

The algorithm used to obtain this output is based on a segment model and searches for people mentions; in this example it marks directly the two entities present in the sentence and outputs them. Like token models, features of chunks are helpful in recognizing entities; two common features are the similarity to an entity in a database and the length of the chunk.

Grammar-based models

A grammar-based model, like context free grammars (CFG) used to specify programming languages, use a set of production rules defined over terminals to express the form of an entity. This is reasonable when the unstructured source requires a better interpretation than the token or segment models; for example, in paper citations, we expect authors, titles and years to be formatted in a certain predefine way; Figure 2.6 below shows this example.

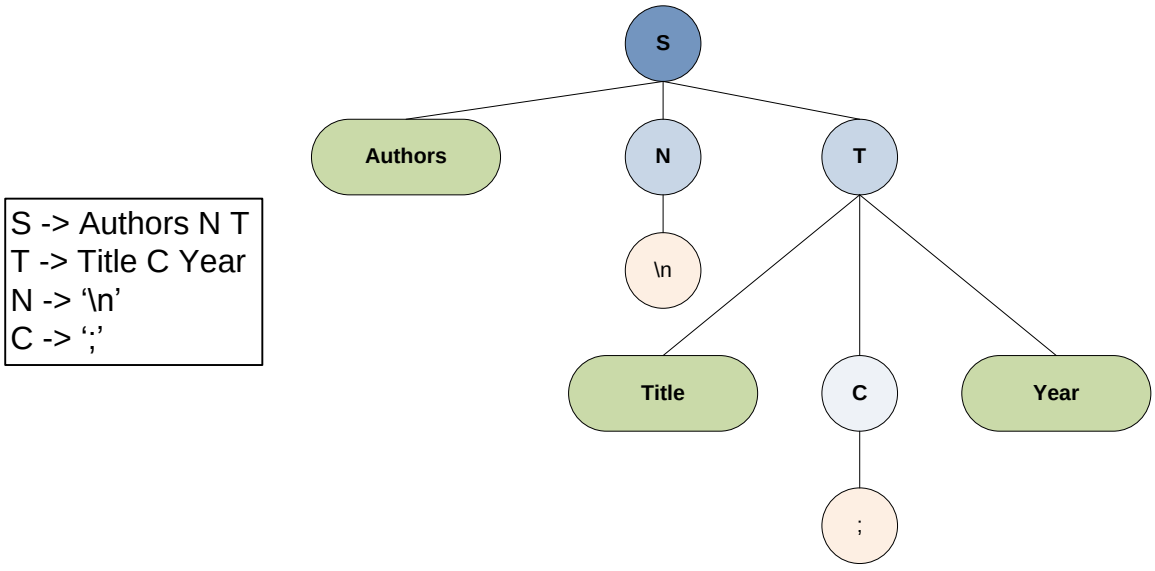

Figure 2.6: A tiny grammar modeling paper citations

The output of an algorithm based on this model will be a list of parse tree, because there will be more than one method to derive entities unlike programming languages, which define strict precedence rules to resolve this issue. The problem becomes to give a score to these parse trees in order to find the optimal, and this can be done in polynomial time; however, because of this, these models are not very diffuse although more expressive than others.

The models described above, as all statistical solutions, need algorithms to be trained and deployed. In the following paragraph we describe a state-of-the-art method for assigning labels to tokens, how to apply it to the models and two strategies, one used for training and one used for inferencing.

Conditional Random Fields

Conditional Random Fields (CRFs) are a class of statistical methods used to model prediction schemes in pattern recognition and machine learning; in statistical NER tasks they can be used to model a single joint distribution $PR(y|x)$ over the predicted labels $y = y_1, ..., y_n$ of the input tokens x. In typical extraction tasks, a chain is suitable to capturing label dependencies and therefore the joint distribution PR can be treated with Markov random fields [10] to express the interdependence between all the elements y_i of the output y; this lead to the conclusion that the label y_i is influenced only by its precedent label y_{i-1}.

Model's probability distribution

State what CRFs are, we report here the conditional distribution of a generic label sequence y over the models, we will denote the dependency between the labels of adjacent tokens by a scoring function $\psi(y_{i-1}, y_i, x, i)$ and we define ψ in terms of weighted function of features as follows:

$$\psi(y_{i-1}, y_i, x, i) = e^{\sum_{k=1}^{K} w_k f_k(y_i, x, y_{i-1})} = e^{w \cdot f(y_i, x, y_{i-1})} \tag{2.1}$$

For the token-level models, with this pre-edge scores, the conditional distribution of a label sequence is:

$$PR(y|x, w) = \frac{1}{Z(x)} \prod_{i=1}^{n} \psi(y_{i-1}, y_i, x, i) = \frac{1}{Z(x)} e^{\sum_{i=1}^{n} w \cdot f(y_i, x, y_{i-1})} \tag{2.2}$$

where $Z(x)$ is a normalizing constant equal to $\sum_y e^{w \cdot f(x,y)}$, the sum of the feature vector $f(x, y) = \sum_{i=1}^{n} f(y_i, x, i, y_{i-1})$ over all tokens positions.

Concerning segment-level models, we can define a probability distribution over a segmentation $s = s_1, ..., s_p$ of the input x as follows:

$$PR(s|x, W) = \frac{1}{Z(x)} e^{w \cdot f(x,s)} \tag{2.3}$$

where $f(x, s) = \sum_{j=1}^{|s|} f(y_j, x, l_j, u_j, y_{j-1})$ is the feature vector (l_j and u_j are respectively the start position and end position of the segment), W is a weight vector for the feature vector f and $Z(x) = \sum_s e^{w \cdot f(x,s)}$ is the normalization term. During the inference phase, the goal will be to find a segmentation $s = s_1, ..., s_p$ of the input $x = x_1, ..., x_n$ such that $w \cdot f(x, s)$ is maximized.
Grammar-based models are scored discriminatively: each production on the form $R \rightarrow R_1 R_2$ is scored as follows:

$$s(R) = s(R_1) + s(R_2) + w \cdot f(R, R_1, R_2, x, l_1, r_1, r_2) \tag{2.4}$$

where (l_1, r_1) delimits the text span for R_1 and $(r_1 + 1, r_2)$ delimits the text span for R_2; therefore the score of a node depends on the text spans of the production and it does not depend on the subtree of children. In the base case (i.e. when a production is terminal), we have that $w \cdot f(R, x, l, r)$ is equal to the score of assigning label R to tokens between the text span delimited as (r, l).

Training and inferencing

Each model outputs labels y for which the score $s(y) = w \cdot f(x, y)$ is maximum, $f(x, y)$ is a feature vector defined jointly over the output y and the input x. For token-level models, y is a sequence of labels, for segment-level models, y is a segmentation of x and for grammar-based models, y is represented with a parse tree.
We denote as $D = \{(x_l, y_l)\}_{l=1}^{N}$ the labeled training set. One strategy used for training is the Likelihood trainer. This method can be used when the term $w \cdot f(x, y)$ is used to impose a probability distribution over the output y as follows:

$$PR(y|x) = \frac{1}{Z(x)} e^{w \cdot f(x,y)} \tag{2.5}$$

where $Z(x) = \sum_y e^{w \cdot f(x,y)}$. Now, the goal during the training phase is to choose a properly weight vector w such that the probability of the correct output, as given in the training set, is maximized. Let $L(w)$ be the logarithm of the probability of the training data with a weight vector denoted as w; we can write it in this way:

$$L(w) = \sum_l log(Pr(y_l|x_l, w)) = \sum_l (w \cdot f(x_l, y_l) - log(Z_w(x_l))). \tag{2.6}$$

At each iteration, the term $-\frac{||w||^2}{C}$ is added to avoid large swings in the parameter values and to perform a soft form of feature selection. The goal is to minimize the number of features and maximize the accuracy, thus the training objective can be expressed as:

$$\max_w \sum_l (w \cdot f(x_l, y_l) - log(Z_w(x_l))) - \frac{||w||^2}{C}. \tag{2.7}$$

which is a concave equation in w and can be maximized by gradient ascendant type of methods.
To compute the inference phase, a common method is to use the Mean Average Precision (MAP) score metric, finding an output for which the weighted feature vector is maximized; more formally we have to find $y^* = \arg\max_y(w \cdot f(x, y))$. The key to solve it efficiently, reducing the number of possible value of y, is that f decomposes over finer substructures in y; this lead to design an algorithm based on the following principle: let the output be S and S_1, S_2 two disjoint decomposition of S, i.e. $S_1 \cap S_2 = \emptyset$ and $S_1 \cup S_2 = S$. Let $S_3 \subset S_1$ be the smallest part of S_1 such that there are no features in S_3 involving a label in $S_1 \setminus S_3$ and S_2. We can define the MAP score as follows:

$$\mathcal{V}(S) = \max_{l \in S_3} \mathcal{V}(S_1|S_3 = l) + \mathcal{V}(S_2|S_3 = l). \tag{2.8}$$

As long as the number of possible labels in S_3 is small, the equation (2.8) will be efficient.

2.2.4 Evaluation

The basic idea to evaluate a NER system is to compare its output with a golden standard collection. Usually, this collection is hand tagged by persons which are not involved in the development of the system, in this manner there can't be prejudices in the way an unstructured document is tagged. Standard metrics [11] to take into account are precision, recall, F-measure and accuracy, that we are going to introduce here. In the context of the classification problems, we are in the situation of Figure 2.7.
We have four types of results:

- True Positive (TP): correct results, in our case these are the entities automatically tagged like the golden standard;
- False Positive (FP): unexpected results, they are the entities automatically tagged which do not appear in the golden standard;

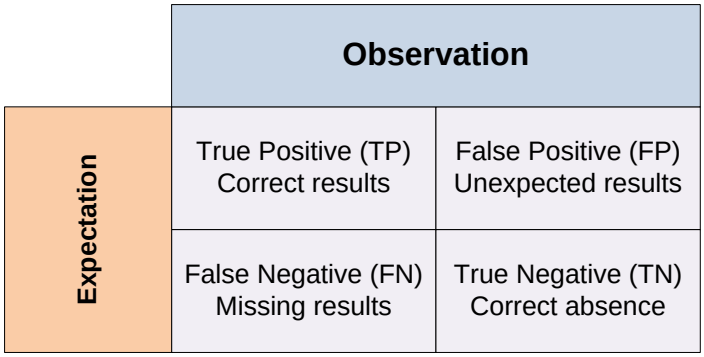

Figure 2.7: Results of a classification problem

- False Negative (FN): missing results, they are entities that appear in the golden standard and miss in the output results;
- True Negative (TN): correct absence, they are items which do not appear both in the golden standard and in the output results.

State this, we can now define the precision score as:

$$P = \frac{TP}{TP + FP}, \tag{2.9}$$

recall as

$$P = \frac{TP}{TP + FN}, \tag{2.10}$$

F-measure (also known as F1 score) as

$$F1 = 2\frac{P \cdot R}{P + R}, \tag{2.11}$$

and the accuracy as

$$A = \frac{TP + TN}{TP + FP + FN + TN}. \tag{2.12}$$

2.3 Relation Extraction

In this section we generalize the RE problem and we describe principal supervised and semi-supervised methods to address it. We also define how n-ary relations can be found and how to evaluate an RE system.

The next level of extraction, after NER, is called *Relation Extraction* (RE). The aim of this task is to encase two or more extracted entities in a semantic relationship; usually, RE systems are focused on binary relations. More formally, the RE task can be specified as:

RE PROBLEM: Let D denote a generic text document containing n entities, $e_i, 0 < i \leq n$ denotes a generic entity, and C the set of m entity classes, c_j, $0 < j \leq m$ denotes a generic class. Let $E = \{(e_i, c_j)|e_i \in D, c_j \in C\}$ be the set of classified entities extracted from D. A relation between entities is formed as a tuple $(e_0, ..., e_k)_\rho$ where entities in it are connected by a semantic relation $\rho = r(c_0, ..., c_k)$ defined over the corresponding classes of the entities inside the tuple. Goal is to find all instances of ρ over D.

Recalling the example of Figure 2.1, we can consider the RE problem in these terms:

D = "**NASA**, the **Russian Federal Space Agency** and their international partners have selected two veteran spacefarers for a one-year mission aboard the **International Space Station** in 2015. This mission will include collecting scientific data important to future human exploration of our solar system. **NASA** has selected **Scott Kelly** and **Roscosmos** has chosen **Mikhail Kornienkov**."
$C = \{Company, Person, Place\}$

The set of classified entities E that D contains can be extracted with a NER system through strategies we specified in the previous section, therefore:

$E =$ { (Nasa, $Company$), (Russian Federal Space Agency, $Company$), (International space station, $Place$), (Scott Kelly, $Person$), (Roscosmos, $Company$), (Mikhail Kornienkov, $Person$) }

If we are interested in understanding which people are hired from a company, we can specify the semantic relation as:

$$\rho = \text{Hired(Company, Person)}$$

A RE system with these inputs should return (Nasa, Scott Kelly)$_{Hired}$ and (Roscosmos, Mikhail Kornienkov)$_{Hired}$ as instances of the relation ρ.

2.3.1 Supervised methods

When enough labeled examples for a specific domain can be found, supervised methods turned out to be the best solution. To be as much clear as possible, in this section we will focus on binary relations, and because supervised methods set the RE problem as a classification problem, we need redefine it as:

RE PROBLEM (CLASSIFICATION): Given a sentence $S = x_1, x_2, ..., e_1, ..., e_2, ...x_n$, where e_1 and e_2 identify two entities with $x_i, 0 < i \leq n$ representing a generic word, and given a relation $\rho = r(c_1, c_2)$ between the respective classes of e_1 and e_2, the goal is to define a function f_ρ such that:

$$f_\rho(F(S)) = \begin{cases} +1 & \exists(e_1, e_2)_\rho \\ -1 & otherwise \end{cases}$$

where $F(S)$ is a feature vector extracted from S.

The function f_ρ can be constructed with classifiers when positive and negative examples are available for their training, using a set of features like POS tagging, orthography and punctuation. In the following two paragraphs we present the two principal supervised approaches.

Feature based Methods

These methods extract from the labeled examples (positive and negative) semantic and syntactic features, such as:

- strings representing the entities;
- classes of the entities;
- string occurring between the two entities;
- string occurring before the first entity and after the second entity;
- number of words between the two entities;

- number of words before the first entity and after the second entity;
- orthography of entities.

These feature are used to train a classifier over the labeled example that will be used to find relations between entities in other documents. The choice of which features must be selected to train the classifier depends on the nature of the problem and in general it's hard to reach an optimal subset of relevant features.

Kernel based Methods

Before defining the two major kernel approaches, we define what string kernel methods are in the context of text classification, following the paper [12]. Given two strings, say x and y, for string kernel methods their similarity score is computed as the number of substrings that are common between them. Each string can be mapped to an higher dimensional space where each dimension correspond to the presence (weighted) or the absence (0) of a particular subsequence. In example, the string *bar* is expressed as follows:

$$\begin{aligned}\phi(x = bar) &= [\phi_a(x) \;..\; \phi_b(x) \;..\; \phi_r(x) \;..\; \phi_{ar}(x) \;..\; \phi_{ba}(x) \;..\; \phi_{br}(x) \;..\; \phi_{bar}(x)] \\ &= [\lambda \;..\; \lambda \;..\; \lambda \;..\; \lambda^2 \;..\; \lambda^2 \;..\; \lambda^2 \;..\; \lambda^3]\end{aligned}$$

where $\lambda \in (0, 1]$ is a parameter that decreases for longer and non-continuous substrings: $\phi_{br}(bar)$ has a weight of λ^3 and therefore is penalized more than $\phi_{ar}(x)$ and $\phi_{ba}(x)$ (both with a weight of λ^2) because br does not occur continuously in *bar*.
More formally, let z be a contiguous substring of x find at indexes $i = i_1, i_2, ..., i_{|z|}$ with $i_1 \leq i_2 \leq ... \leq i_{|z|}$ denoted as $z = x[i]$ and let $l(i) = i_{|z|} - i_1 + 1$ be the length of z. Obviously, z can be inside x in various ways and x's coordinate that collocate z in the higher dimensional space is equal to:

$$\phi_z(x) = \sum_{z=x[i]} \lambda^{l(i)}. \tag{2.13}$$

If Z is the set of all possible combination of substrings which can be found in x and y, the kernel similarity between the two strings is given by the equation:

$$K(x, y) = \phi(x)^T \phi(y) = \sum_{z \in Z} \phi_z(x)^T \phi_z(y) \tag{2.14}$$

The equation 2.14 lead to an exponential complexity; however it can be computed efficiently using a dynamic programming strategy described in [12].

The function $K(x, y)$ previously described can compute the similarity between two strings, but it can compare also objects like parse trees and whole sentences. In RE, if we represent as x^+ a positive sample and as x^- a negative sample, $K(x^+, y) > K(x^-, y)$ implies that the first contains a valid instance of the relation. The kernel function K is used to train logical classifiers over sentences and the objects x^+ and x^- can be modeled as parse trees containing the sentence in which entities are found or words around the two entities; due to those two possibilities there exist two main kernel approaches: Bag of features Kernel and Tree Kernel.

Bag of features Kernel

The idea of this kernel method is that the context around the entities it's sufficient to determine if they are in a certain relation or not. If in a general sentence $S = x_1, x_2, ..., e_1, ..., e_2, ..., x_n$ entities e_1 and e_2 are in relation, we can derive another representation of $S = [prefix, e_1, middle, e_2, suffix]$ where prefix, middle and suffix are the word context at the relative position from the two entities in the sentence. When we find a new sentence S' with entities e'_1 and e'_2, the similarity between the two sentences S and S' is computed using the equation 2.14 with their prefix, middle and suffix portions working by word level: if the similarity score is high, entities e'_1 and e'_2 will form a new relation. If we want to define three different subkernels, one for each portion, the final kernel will be a sum of them [13].

Tree Kernels

Instead of considering strings, this approach relies on shoal parser trees constructed over the sentences: it exploits the robust representation of parser trees to perform the RE task. For each sentence containing two entities in relation its parser tree is computed; afterwards subtrees which cover the two entities are numbered as positive sample while subtrees not covering both entities are numbered as negative samples. Figure 2.8 shows in part (a) a parse tree, in part (b) a positive sample and in part (c) a negative sample.
Given two shoal parser trees referring to two sentences, the kernel function is used to compute their similarity regarding their own structures by calculating a weighted sum of the number of subtrees the sequences have in common; therefore, the kernel function can be redefined with a recursive algorithm [14] taking as input two subtrees rooted at T_1 and T_2:

$$K(T_1, T_2) = \begin{cases} 1+K(child(T_1), child(T_2)) & F(T_1) = F(T_2) \\ 0 & \text{otherwise} \end{cases} \tag{2.15}$$

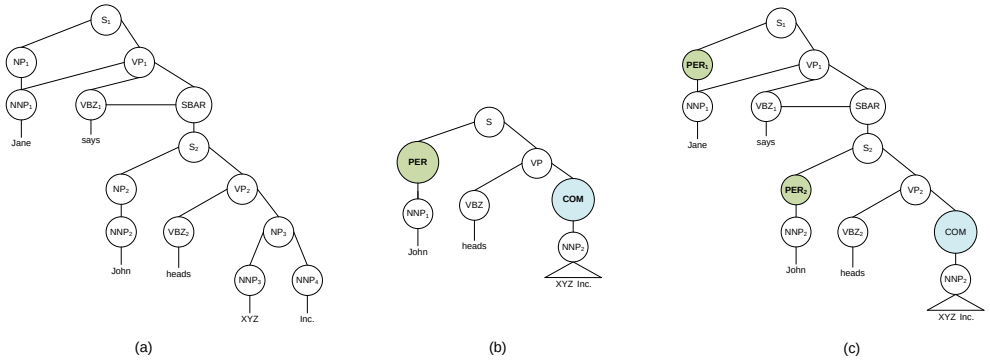

Figure 2.8: A parse tree with positive and negative samples

where $child(T_1), child(T_2)$ and $F(T_1), F(T_2)$ are the respective children sequences and attributes (POS tag, word, etc.) of the corresponding nodes T_1 and T_2.
Like the bag of feature kernel approach, if the score of this new similarity function reasonable, entities involved in that subtree will form a new relation.

Shortest path

Sometimes the predicate between two entities in a sentence it's enough to determine both if there is a relation between entities and the nature of the relation. Sentences are scanned with a dependency parser generating a dependency parser tree, and then the shortest path between entities is computed. Let $P = e_1 \rightarrow w_1 \rightarrow ... \rightarrow w_i \leftarrow ... \leftarrow w_n \leftarrow e_2$ be the shortest path between two entities e_1 and e_2, w_i indicates words in the shortest path and arrows the direction of the dependency. Due to the sparsity of data, features are extracted from each word, generating feature vectors as Cartesian products in this form:

$$x = [f(e_1)] \times [\rightarrow] \times [f(w_1)] \times [\rightarrow] \times ... \times [f(w_i)] \times ... \times [\leftarrow] \times [f(w_n)] \times [\leftarrow] \times [f(e_2)]$$

where $f(w_i)$ is the features extracted from w_i. We need also to redefine the kernel function as:

$$K(x, y) = \begin{cases} 0 & \text{if } |x| \neq |y| \\ \prod_{i=1}^{|x|} f_c(x_i, y_i) & \text{otherwise} \end{cases} \tag{2.16}$$

where $f_c(x_i, y_i)$ is the number of features x_i shares with y_i.
Drawbacks of all supervised methods described above are:

- if we want to find an higher order relation than the binary, we need to redefine all equations described above;
- they are hard to extend to a new relation type since there they need labeled data;
- they require to pre-process input data and this can lead to error propagation.

2.3.2 Semi-supervised methods

For some domains, find labeled data to be used to train classifiers used in supervised methods is almost hard. Semi-supervised methods try to infer relations exploiting the context of entities. Consider those documents:

(1) "When **eBay** *founder* **Pierre Omidyar** was 31 years old, he became a billionaire in one afternoon.",
(2) "**Mark Zuckerberg** *is the CEO of* **Facebook**, which he started in his college dorm room in 2004.",
(3) "**Steven Paul Jobs** *was born in* **San Francisco** *on* **February 24, 1955**.";

if we collect suggestions given by Bag of feature Kernel approach described in section 2.3.1, entities context in the above documents can be sufficient to determine respectively those relations:

(1) **founder**(Pierre Omidyar, eBay)
(2) **is-CEO-of**(Mark Zuckerberg, Facebook)
(3) **born**(Steven Paul Jobs, San Francisco, February 24, 1955).

We present here three state-of-the-art approaches following this strategy, the first two work over a specific relation and a seed set both given by hand, the last tries to learn relations by its own in a self-supervised way.
We first focus on Yarowsky algorithm [16], because the first two methods we are going to introduce rely on this. The idea of this algorithm is to use the output of weak learners as training data for the next iteration, the seed set to be used is given by hand. The algorithm, originally used for word sense disambiguation, follows those steps:

Yarowsky

```
1 Collect a set of unlabeled data D;
2 Find a set of seed examples S from D;
3  repeat
4   Train a classifier C on S;
5   Label D using C;
6   N = top n labels in which C is highly confident;
7   S = S ∪ N;
8   D = D \ N;
9 until a convergence criteria is reached
```

Dual Iterative Pattern Relation Extraction (DIPRE)

Starting from a small seed of instances, the DIPRE method [17] follows the Yarowsky algorithm and is able to expand the seed set in order to find other instances of a given binary relation over a collection of documents D. We can summarize how it operates with the following algorithm:

Dipre

```
1 Give a seed set containing searched relation's tuples;
2 Find tuple's occurrences and contexts in the collection;
3 Generate patterns by the occurrences with similar context;
4 Use patterns generated to find other occurrences;
5 Return to step 2 unless a convergence criteria is reached
```

In the second step, the context of each tuple is defined as another tuple of six elements, in the form:

$$[\text{order}, e_1, e_2, \text{prefix}, \text{suffix}, \text{middle}]$$

where:

- e_1 is the first entity of the relation;
- e_2 is the second entity of the relation;
- *order* is 1 if e_1 occurs before e_2, 0 otherwise;
- *prefix* is represented by the ten characters before the match;
- *suffix* is represented by the ten characters after the match;
- *middle* is represented by the string between e_1 and e_2.

The hardest step is the third, where given the tuples with their context, we want to generate patterns from them. DIPRE groups tuples by parameters *order* and *middle*; afterwards, for each group, the longest common suffix and prefix is computed, and the final tuple

$$[\text{order}, e_1, e_2, \text{longest common prefix}, \text{longest common suffix}, \text{middle}]$$

represents a new pattern which can be used in the fourth step.
For example, let us search for the relation $wrote(author, book)$ with the initial seed set $S = \{(\text{Dan Brown, The Da Vinci Code})\}$, and suppose we consider

the following documents:

(1) "Read **The Da Vinci Code** by **Dan Brown** online from this resource."
(2) "The famous thriller writer **Dan Brown** wrote **The Da Vinci Code** in 2003."
(3) "When the writer **Dan Brown** wrote **The Da Vinci Code** in 2003 he didn't want to attack the Roman Catholic Church."

They will lead respectively to the following contextualizing tuples:

(1) [0, Dan Brown, The Da Vinci Code, Read, online from, by]
(2) [1, Dan Brown, The Da Vinci Code, thriller writer, in 2003., wrote]
(3) [1, Dan Brown, The Da Vinci Code, the writer, in 2003 he, wrote];

the first tuple will not match any other tuple while the second and the third are grouped; together they will induce those two new patterns:

(1) [0, author, book, Read, online from, by]
(2,3) [1, author, book, writer, in 2003, wrote]

that will be used in the next iteration to find more occurrences of the given relation; for example the pattern (2,3) given the document

"The fantasy writer **Joanne K. Rowling** wrote **Harry Potter and the Order of the Phoenix** in 2003 after three years from the release of her last Harry Potter novel."

will find the tuple (Joanne K. Rowling, Harry Potter and the Order of the Phoenix).
An important component of the system is composed by the initial seed set, that must be chosen properly. Another key element is the stopping criteria, for example it can be "no more occurrences were found" or "the number of occurrences has exceed a certain threshold".

Snowball

One of the drawbacks of DIPRE is that when patterns are generated, tuples are grouped with an exact match. This means that even if the suffix or prefix of a tuple differ by a single punctuation to the ones of another tuple, the two tuples are not matched together. The "Snowball" system [18] it's been developed to overcome this issue, and gives better results than DIPRE.

Snowball can be described by the architecture showed in the Figure 2.9 below:

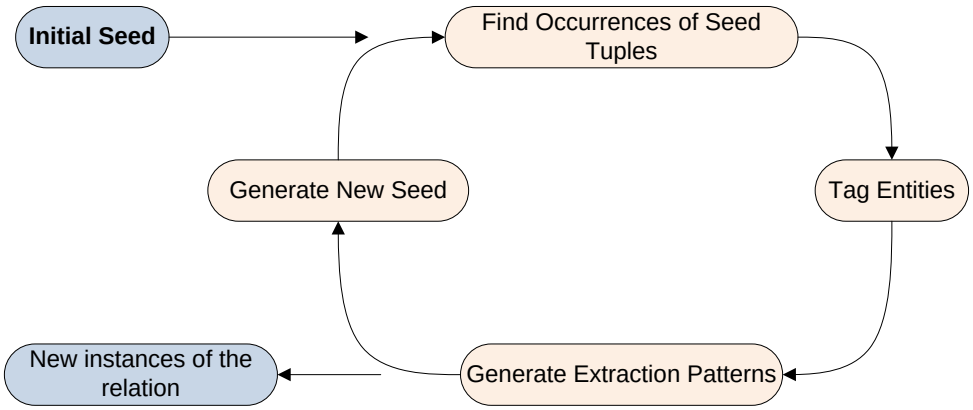

Figure 2.9: Architecture of Snowball

Like DIPRE, the algorithm starts with an initial seed of instances of the binary relation searched, say for example ρ = located(Company, Location). Then, it finds occurrences of the seed instances in the collection, and use a NER system to tag entities in order to avoid spurious matches; each occurrence is converted into a tuple containing five elements:

$$[\text{prefix}, c_1, \text{middle}, c_2, \text{suffix}]$$

where:

- c_1 is the class of the first entity of the relation, in our case equal to *Company*;
- c_2 is the class of the second entity of the relation, in our case equal to *Location*;
- *prefix*, *suffix* and *middle* are represented like the vector space model in IR-fashion: they are three vector associating weights with words.

Recalling our example, if in the seed set we have (Apple, Cupertino) and we find a document such as "... and he went to Apple campus in Cupertino to develop ...", Snowball will generate the tuple:

[((*p*$_1$, went), (*p*$_2$, to)) , *Company*, ((*m*$_1$, campus), (*m*$_2$, in)), *Location*, ((*s*$_1$, to), (*s*$_2$, develop))];

in this case we have a limit of 2 words composing each vector. Each word weight w of the correspondent word in the feature vector v is computed as:

$$weight(w, v) = \frac{tf(w,v)}{|v|}$$

where $tf(w, v)$ is the number of times w occurs in v.
To match tuples, Snowball defines the similarity function *match* between two tuples $t_1 = [p_1, C_1, m_1, C_2, s_1]$ and $t_2 = [p_2, C_1', m_2, C_2', s_2]$ as:

$$match(t_1, t_2) = \begin{cases} p_1.p_2 + m_1.m_2 + s_1.s_2 & C_1 = C_1' \wedge C_2 = C_2' \\ 0 & otherwise \end{cases}$$

Then, Snowball use a clustering algorithm to group tuples using the above function, setting also a minimal similarity. Tuples are collapsed in three centroid vectors $\bar{p_s}, \bar{m_s}, \bar{s_s}$ representing the prefix, middle and suffix of the new pattern:

$$[\bar{p_s}, C_1, \bar{m_s}, C_2, \bar{s_s}],$$

that is used to find new instances of the binary relation and to find other patterns in the next iteration over the collection.
A confidence score is also assigned to each pattern:

$$Conf(P) = \frac{P_{positive}}{(P_{positive}+P_{negative})},$$

where $P_{positive}$ is the number of positive matches and $P_{negative}$ is the number of negative matches. In our example, if the new pattern P finds those three instance: (Apple, Cupertino), (Microsoft, Redmond) and (Apple, Chicago), the last instance is counted as negative match because there is still an instance (Apple, Cupertino) in the seed set, and $conf(P) = \frac{1}{1+1} = 50\%$. This score is used to give more importance to instances extracted from patterns with higher score than the others.

TextRunner

Unlike DIPRE and Snowball systems which work on a user defined relation and require an initial seed, properly chosen, to expand instances of the given relation, the "TextRunner" [19] system learns entities, classes and relations without any human input, and it is composed by three key modules:

- Self-Supervised Learner: given a non-tagged corpus, this component outputs a classifier which tags candidate instances of extraction as reliable or not.
- Single-Pass Extractor: it does not use a parser, but it extracts each candidate from the corpus and sends it to the classifier, retaining candidates labeled as reliable.
- Redundancy-Based Assessor: it assigns a probability to each retained candidate on the base of a probabilistic model of redundancy in text.

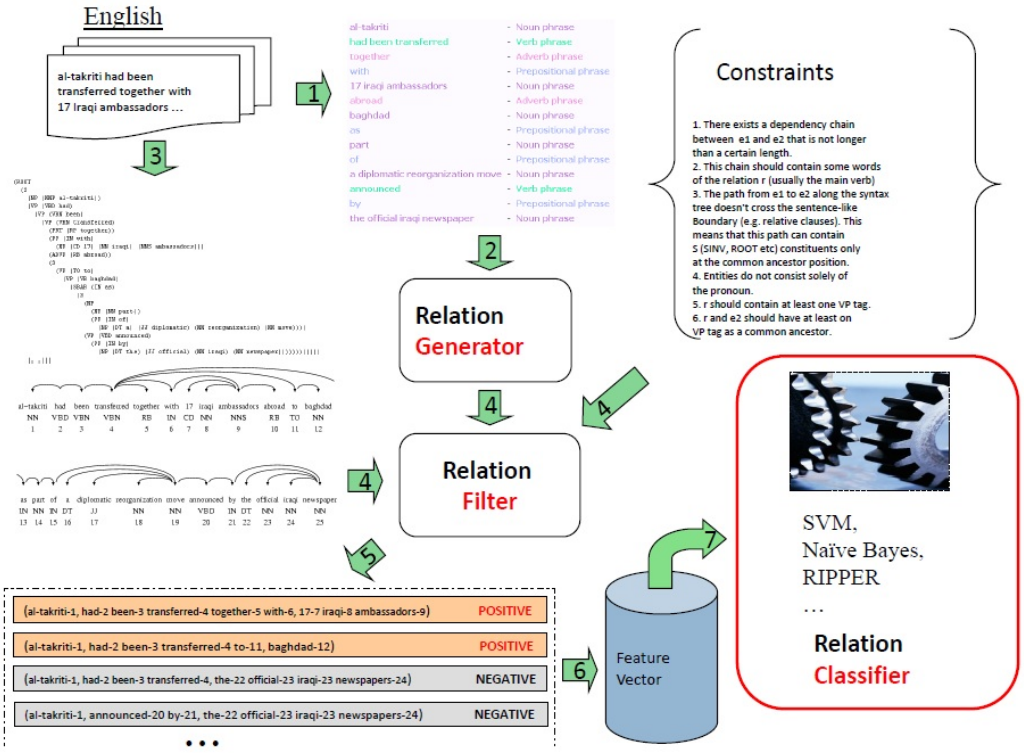

Figure 2.10: Steps for training TextRunner's self-supervised learner

Figure 2.10 shows in details the steps used to train the learner:

1. a chuncker is invoked over the small training corpus;
2. a relation candidator generate possible relations;
3. a parser is used to tag the training corpus and find possible relations;

4. the relation candidator and the parser form the set of candidate relations;

5. the candidate relations are marked as positive or negative samples using parser trees, dependency trees and some constraints;

6. each relation is mapped into a feature vector;

7. the labeled feature vectors are used like training set for a logical classifier.

2.3.3 N-ary relations

In previous sections we focused on binary relation when we described methods for RE; this is convenient for explaining method's concepts but, in a real world scenario, we are often interested in relations with higher degree. For example, bio-medical tasks require to extract a 4-ary relation to describe the mutation of a disease, the relation describing where and when a person was born is a 3-ary relation like (Person, Place, Data) and so on.
A naive approach to find high order relationships is to list all the tuples to train a binary classifier on positive and negative examples, but the problem is the candidate size that will grow exponentially: for a relation connecting n entities with each element having m possible way, there are $O(m^n)$ possible relation candidates.
A graph-based approach described in [15] does not require to list all tuples and follows those steps:

1. Recognize binary relations which are arguments of the relation of interest.

2. Entities involved in a relation can be viewed as nodes connected by an edge representing the relation.

3. Maximal cliques in the constructed graph will construct complex relations.

Moreover, a weight score is assigned to each edge of the graph corresponding to the probability that the two connected entities are in relation according to the classifier. Then, the geometric mean is used to give a confidence score of the maximal clique:

$$w(C) = (\prod_{e \in E_C} w(e))^{\frac{1}{|E_C|}}, \tag{2.17}$$

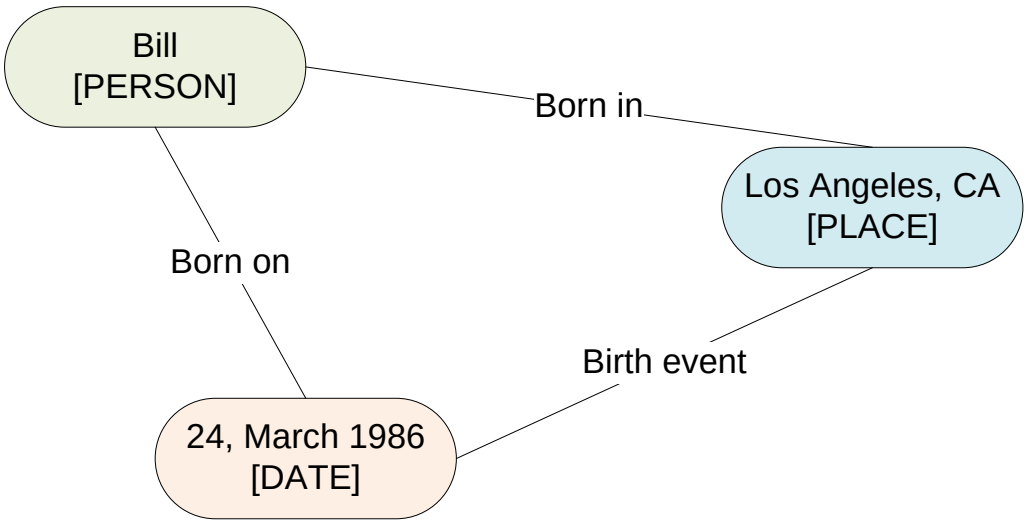

Figure 2.11: A graph constructed to detect a 3-ary relation

this score is interpreted as the confidence for each possible complex relation. A threshold is fixed and only maximal cliques with a confidence score grater than this are considered as valid relations. This approach it's the preferred because it's based on binary relation which have well tested methods of creation and the number of binary relation is much smaller than the number of possible complex relations of the naive approach.

2.3.4 Evaluation

The techniques to evaluate an RE system depends on the method implemented, therefore we present two strategies for evaluation: one for supervised methods and one for semi-supervised methods.

Like the NER task, we can see supervised methods in RE like a classification problem and therefore use the same measures described in section 2.2.4 to evaluate a system developed with this approach. The number of correct relations extracted is determined through labeled data required for this method, thus we can define precision P, recall R and F-measure $F1$ as:

$$P = \frac{\text{number of correct relations extracted}}{\text{total number of relations extracted}} \tag{2.18}$$

$$R = \frac{\text{number of correct relations extracted}}{\text{actual number of relations extracted}} \tag{2.19}$$

$$F1 = 2\frac{P \cdot R}{P + R} \tag{2.20}$$

In semi-supervised approach, labeled data is not available and we can't use equations defined above, but the measures remain the same. A common practice is to select randomly a sample of the system's output and this sample it's checked by hand in order to find correct relations. Getting an estimation of the recall measure is more difficult than the precision because those methods are often applied to large amounts of data and it's hard to obtain the exact number of relations in it.

Chapter 3

Real world scenario

In this chapter we describe the real word scenario which brought us to define our novel approach in RE. As first, we give a general introduction of the company which proposed the problem, then we describe how company's editorial division currently works and finally give a formal definition of the problem to be solved.

3.1 Company profile

2night[1] is one of the leader company in Italy about information and multichannel communication of the entertainment sector. It was founded on the idea of transfer American information models in Italy and the original project was launched in March of 2011; currently it involves four channels of communications:

- online portal, covering thirtytwo districts in Italy;
- magazines, distributed in main locations;
- mobile, with an android and and iOS application;
- radio, having a partnership with local broadcasters.

2night concentrate reviews and programs of most famous italian locals, concerts, theater shows and parties allowing people to be updated on the main entertainment events of their city, answering the question "what's hot today?".

[1] www.2night.it

3.2 Editorial office and press reviews

One of the most important department of *2night* is the editorial division, that analyze and publish new events for the company web site and the magazines. Currently, *2night*'s editorial division relies on affiliate news agencies which submit press reviews, i.e. unstructured documents of text verbosely describing entertainment events. The journalist working in the editorial division, when receiving the press reviews, identifies events reading the whole document, inserts new entries in the event's database and finally publishes an article on the company's site talking about the events found.
The event's database is a relational table used implicitly by web site user's at the time of searching for events in a certain city or area. The search generates a query which runs over the database and returns the correspondent entertainment events of the city the user selected.

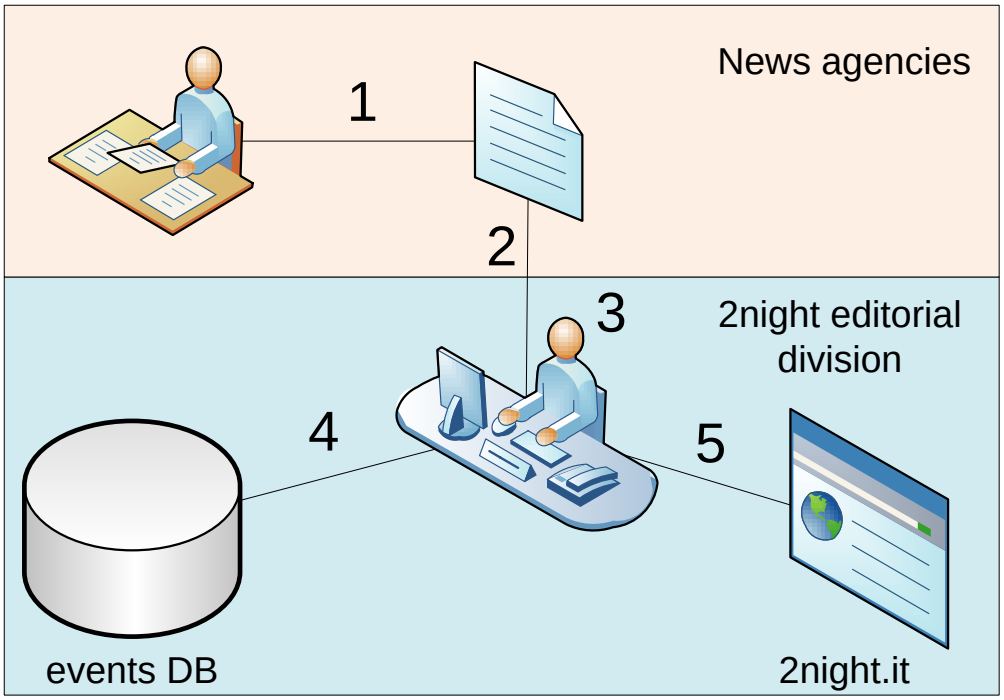

Figure 3.1: *2night*'s information flow

Figure 3.1 present the information flow of the operation described above:

1. A news agency composes a press review;
2. The press review is sent to 2night's editorial division;

3. A journalist reads and analyzes the unstructured document;

4. New entries correspondent to new entertainment events are added to the events database;

5. The journalist writes an article on events he has detected in *2night*'s site.

We present here a press review that will be taken as example when we will introduce core steps of our method used to resolve the problem.

```
Il 2011 e' stato il suo anno.
L'omonimo album di debutto lha resa celebre in ogni dove coronandola la
nuova musa made in UK.
Un grande successo di pubblico e critica ottenuto grazie alla vincente
combinazione di bravura, classe e passione che Anna Calvi riesce ad
esprimere con la sua musica e attraverso i live show. Anna Calvi e' una
grande artista, una fuoriclasse. Gia' indaffarata per i prossimi show
estivi che la vedranno ospite di numerosi ed importanti festival,
Anna Calvi fara' tappa in Italia per due date:
Martedi 24 Luglio
Roma  Parco di San Sebastiano
Roma Vintage
Via di Porta San Sebastiano 2 (P.le Numa Pompilio), 00187 Roma
Biglietto: 15,00 euro + d.p.
Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
Giovedi 26 Luglio
Bologna - Bolognetti Rocks
Rocker Festival
Vicolo Bolognetti, 40125 Bologna
Biglietto: 10,00 euro + d.p.
Prevendite attive da Lunedi' 14 Maggio sui circuiti vivaticket.it,
ticketone.it. I singoli Blackout, Desire, Jezebel, Suzanne and I sono
stati accolti con ottime critiche e hanno  riscosso fin da subito
successo e curiosita'. Tutti racchiusi nellalbum di debutto, uscito
per la Domino Records, sono accompagnati da video ricchi di charme
e pathos. Le cover di ogni singolo sono state realizzate con foto e
immagini create da Anna Calvi e Emma Nathan. Anna Calvi: per essere
un album di debutto e' un capolavoro, una magistrale scomposizione
di desiderio e amore. Si sviluppa sulla straordinaria chitarra di Anna
e sulla sua potente e ammaliante voce; e' un album indimenticabile e
appassionante. Influenzata dalle vocalita' di artisti diversi come
Nina Simone, Maria Callas e Scott Walker, dalle chitarre di Django
Rheinhard e Robert Johnson, dal classico romanticismo di Ravel e
Debussy, Anna Calvi anche se ispirata da musicisti di un lontano passato,
ha un sound totalmente attuale ma  soprattutto originale. Complici lo
sguardo ipnotico e una bellezza sensuale, Anna Calvi ha conquistato le
copertine ed intere pagine delle migliori riviste e magazine francesi,
tedeschi ed Italiani.
```

```
39 Per i suoi concerti Anna Calvi ha ingaggiato una intima band di appoggio
40 formata da Mally Harpaz alla chitarra, armonium e percussioni, e Daniel
41 Maiden alla batteria, e i risultati sono spesso straordinariamente epici.
42 Benvenuti nel magico mondo di  Anna Calvi   un luogo dove  bellezza e
43 oscurita' complottano e si scontrano tra loro, dove indomite emozioni
44 conquistano e consumano.
```

Due to the unstructured nature of press reviews, find events hidden in them becomes hard even for journalists. For this reason, it's been chosen to give an automatic help in understanding events hidden in those kind of documents.

3.3 Definition of the problem

Before introducing our problem, we need to define what's an entertainment event. An entertainment event can be described as a tuple of three entities of the classes *Date/Range, Location/Place, Artist* meaning that an artist performs in a certain location or place on a certain date or set of dates. As an instance, our example of press review defined in the previous section contains two entertainment events which can be understood by reading rows 9-20 and can be represented by tuples:

1. (Martedí 24 luglio, Roma, Anna Calvi)
2. (Giovedí 26 luglio, Bologna, Anna Calvi)

We are interested in finding triples corresponding to entertainment events in press reviews to help editorial journalist of the company in adding correct entries in the events database. Let's define formally our scenario:

ENTERTAINMENT EVENT DETECTION PROBLEM: Let P denote a generic press review of unstructured text containing a set of entities E of the classes $C = \{Date, Range, Location, Place, Artist\}$. We want to return all valid instances of relations:

- $perform_1(Date, Location, Artist)$
- $perform_2(Date, Place, Artist)$,
- $perform_3(Range, Location, Artist)$,
- $perform_4(Range, Place, Artist)$

which denote entertainment events. We will use this notation $perform(Date/Range, Location/Place, Artist)$ to indicate all four relations.

Obviously our problem shifts to NER and RE tasks, because we want to find all instance of 3-ary relations between certain classes of entities from an unstructured document. Next section will describe how we faced this problem and motivations on the strategy implemented.

Chapter 4

Approaches and implementation

In this chapter we describe the main approach we took addressing our problem. In the first section we define our main strategy, the motivation which lead us to formulate it, the system design and the implementing language. In the second section the Named Entity Recognizer is deeply explained in all its modules, while in the last section the Relation Extractor is fully documented.

4.1 Strategy

Given the problem defined in section 3.3, concretely we have to find in an unstructured document some precise relations between predefined entities of certain classes. All the methods described in chapter 2 have one principal drawback: they exclusively use a sentence-level approach over the lone unstructured document to perform their task. Relations, especially in our type of documents, can span over the single sentence. Consider as an example this piece from the press review reported in section 3.2:

```
1  Il 2011 e' stato il suo anno.
2  L'omonimo album di debutto lha resa celebre in ogni dove coronandola la
3  nuova musa made in UK.
4  Un grande successo di pubblico e critica ottenuto grazie alla vincente
5  combinazione di bravura, classe e passione che Anna Calvi riesce ad
6  esprimere con la sua musica e attraverso i live show. Anna Calvi e' una
7  grande artista, una fuoriclasse. Gia' indaffarata per i prossimi show
8  estivi che la vedranno ospite di numerosi ed importanti festival,
9  Anna Calvi fara' tappa in Italia per due date:
10 Martedi 24 Luglio
11 Roma  Parco di San Sebastiano
12 Roma Vintage
```

```
13 Via di Porta San Sebastiano 2 (P.le Numa Pompilio), 00187 Roma
14 Biglietto: 15,00 euro + d.p.
15 Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
16 Giovedi 26 Luglio
17 Bologna - Bolognetti Rocks
18 Rocker Festival
19 Vicolo Bolognetti, 40125 Bologna
20 Biglietto: 10,00 euro + d.p.
```

The two entity relations forming two entertainment events occur in more than one sentence:

1. (Martedí 24 luglio, Roma, Anna Calvi) → rows 9, 10 and 11,
2. (Giovedí 26 luglio, Bologna, Anna Calvi) → rows 9, 16 and 17.

It's understandable that current state-of-the-art methods can't detect a relation spanned in more than one sentence like (1) and (2) as they all work by sentence-level.
Moreover, relations can span also outside the single document: think beyond semantic relations and image we have a lot of documents about those two entertainment events. Those documents can be for example *tweets* from Twitter, changes of status on Facebook or simply paragraphs of blogs. Figure 4.1 shows a *tweet* speaking about the first event, we can simply find in it the correspondent date (24 Jul), the place (Roma) and the artist (Anna Calvi). Figure 4.2 report the title of a news in a blog, and we can find all the three entities we need to compose the first relation. Finally, Figure 4.3 shows a status on Facebook where we can easily deduct the characteristics of the first event.

Figure 4.1: A *tweet* about our sample event

Press reviews are often verbose and one can often find not only the entities corresponding to the event described, but other entities of the same classes; this create a lot of confusion.
Learned current limits, we subdivided our new method in two main parts, (1) entities are detected in the unstructured press review, (2) starting from

Figure 4.2: A blog title about our sample event

Figure 4.3: A facebook post about our sample event

the extracted entities of the first part, candidate tuples are extracted. Afterwards, a *Fresh Social Knowledge* (FSK) will be used to retrieve other documents concerning the candidate entertainment events and these documents will be analyzed and used to rank candidate tuples. If documents about the event we are searching are abundant, this can be sufficient to distinguish valid candidate tuples from invalid ones and therefore infer the right entertainment events hidden in the press review. All these aspects give to our

approach a social appearance rather than the static one of the well-known methods: the more people write about the entertainment event in the Web, the more will be easy to detect it.

4.1.1 SEED - Social Entertainment Events Detection

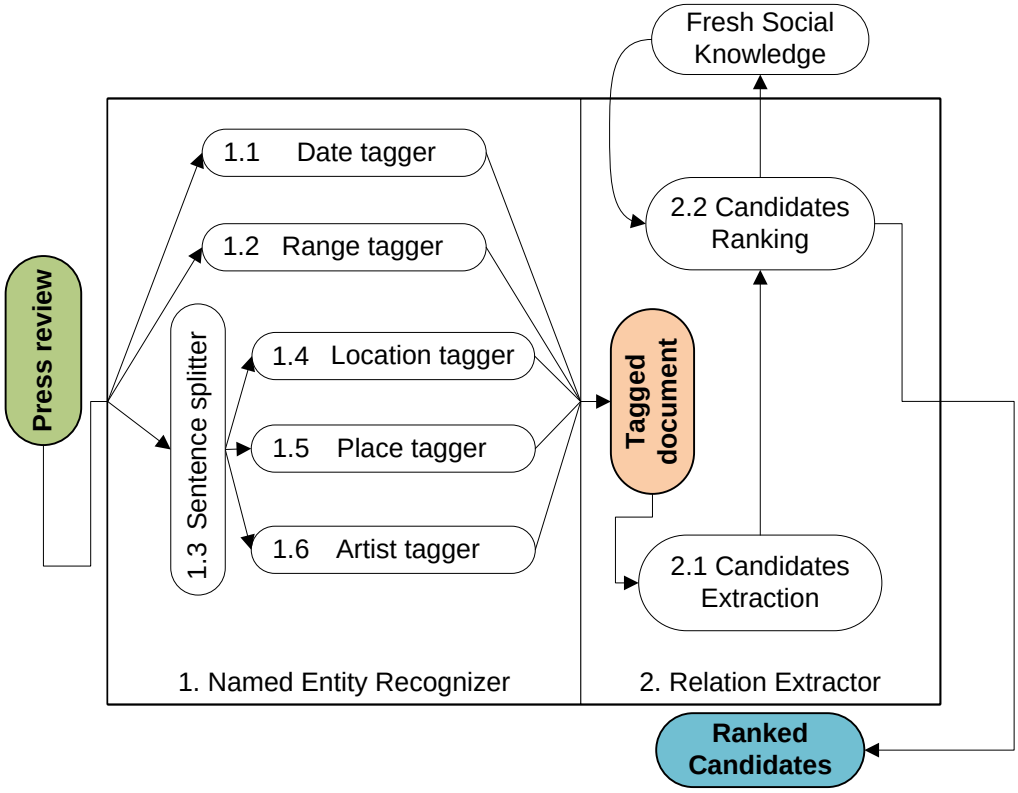

Figure 4.4: SEED architecture

We developed a framework called *Social Entertainment Event Detection* (SEED) in application of our approach, Figure 4.4 shows its overall architecture. As we can see, starting from the unstructured press review, a tagged document is generated passing through the Named Entity Recognizer module. This first module involves five sub-modules (1.1 - 1.6); module 1.3 splits the sentences while modules 1.1, 1.2, 1.4, 1.5 and 1.6 will tag our classes of entities: respectively $Data, Range, Location, PlaceandArtist$. The tagged document which comes out as output of the first phase has the same structure of the press review, but it is enriched with tags that indicates the presence or not of entities we are interested in. Then, the tagged document is used as the input of

the Relation Extractor module. First, the candidates extractor module (2.1) forms candidate tuples of our searched relations. Then, tuples are passed to the candidate ranking (2.2) module that, helped by an external *Fresh Social Knowledge* (FSK), ranks candidate tuples. The output is represented by the list of scored tuples; hopefully the tuples with a score much higher than the others represents valid instances of our relations and lead to the discovery of new entertainment events which were hidden in the starting press review.

4.1.2 Implementing language

One of the most important implementation choices is represented by selecting the appropriate language, which has to permit the execution of the system under different platforms and has to offer the proper data structure and instructions to develop an efficient system.
It has been chosen the *Python*[1] programming language because it offers the principal properties listed above:

- Multiplatform: source code can be utilized under principal operating systems without any modify;
- Easy and direct syntax: develop an entire system with Python is much faster than implementing it with other programming languages;
- Quick: due to the automatic and dynamic management of the memory, its performance are similar to the one of *Java*;
- Open source: there is no need to buy a license to use it.

Python is an interpreted high level scripting language, but the source code is converted in intermediate code called byte-code which is utilized after the first execution of the program, avoiding to interpret the source code each time the program needs to be executed and increasing performance.
In *Natural Language Processing* (NLP), Python is very useful because data structure are suitable for manage also big text file, as Python bufferizes data. It has also a vast and active online community that helps developers and share useful scripts. It exists also a platform, called *Natural Language Toolkit*[2] that includes various libraries that are very useful for writing program involving NLP, and it will be utilized for developing the Named Entity Recognizer module. Furthermore, Python has also a library called *Beautiful Soup*[3] that allows to scrape the Web with few, simple and clear commands.

[1] http://www.python.org/
[2] http://nltk.org/
[3] http://www.crummy.com/software/BeautifulSoup/

4.2 NER - Named Entity Recognizer

In this section we describe the approach taken to develop the fist module of the system SEED, the Named Entity Recognizer. We list motivations which lead us to the method adopted together with six subsection, each one describing one submodule. In order to see how each submodule works, the press review shown in chapter 3 is recalled at the end of each subsection, and the usage of every submodules is explicated with it.

Relations we are interested in are 3-ary tuples composed by entities of the classes $Data/Range, Location/Place, Artist$. By analyzing various press reviews, we arrived to the following conclusions:

- Artists are well known and basically are composed by the name mention of the artist/group band;
- Places can be listed using 2night's places database;
- Locations can be learned from the Italian version of *Wikipedia*[4] which contains a different page for each municipality in Italy;
- Dates and Ranges have predefined forms.

Due to these aspects, we decided to adopt the gazetteer approach for the classes $Location/Place$ and $Artist$, while for the classes $DataandRange$, rule based methods in the form of *Regular Expressions* (REs) will be exploited. The Named Entity Recognizer module will pass to its submodules the unstructured press review and will collect all the tags they generate, forming the tagged document. Entities will be tagged with the following standard form:

- [art NM] for the class $Artist$;
- [loc NM] for the class $Location$;
- [place NM] for the class $Place$;
- [data NM] for the class $Data$;
- [range NM] for the class $Range$,

[4] http://it.wikipedia.org/

where NM denotes a name mention of the correspondent entity. As stated above, the tagged document will be without a predefined structure as the press review, but it will be enriched by those tags. This choice is particularly helpful when we will evaluate this first module of SEED.
Like we explained in section 2.2.1, we use the knowledge based method because we need an high precision score and our domain is closed. However, sometimes ambiguities will rise, but our module will leave them as the task of disambiguation is not crucial for our final scope.

4.2.1 Date tagger

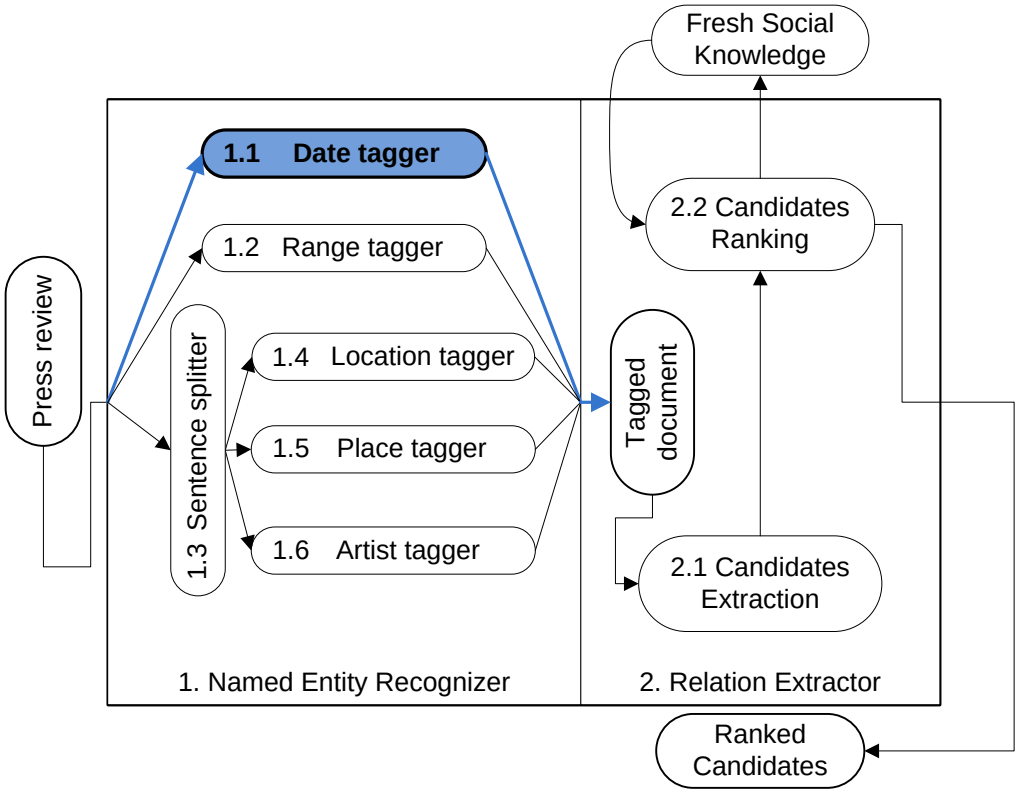

Figure 4.5: Date tagger module

The Data tagger module takes care of tagging the elements of the class *Date*. As described in the introduction of this section, REs are used to detect well-known form of dates.
Before introducing the set of REs used, we need to define the name of the days and months in Italian:

$dayName$ = "lunedi|martedi|mercoledi|giovedi|venerdi|sabato|domenica";
$monthName$ = "gennaio|febbraio|marzo|aprile|maggio|giugno|luglio|agosto|
settembre|ottobre|novembre|dicembre".

Days, months and years can have also a numerical form, expressed bu the following expressions:

$dayNumber$ = 3[01]|[12][0-9]|0?[1-9];
$monthNumber$ = 1[012]|0?[1-9];
$year$ = (19|20)?[0-9][0-9].

Consequently, the element of the set of REs used to detect and tag entities of the class Date are:

$dmy1 = (dayName)?[\ ]?(dayNumber)[\ ](monthNumber|monthName)[\ ](year)$;
$dmy2 = (dayName)?[\ ]?(dayNumber)[/](monthNumber|monthName)[/](year)$;
$dmy3 = (dayName)?[-]?(dayNumber)[-](monthNumber|monthName)[-](year)$;
$dayMonth = (dayName)?[\ ]?(dayNumber)[/\ -](monthNumber|monthName)$;
$dayAlone = (dayName)[\ ](dayNumber)$.

However, even if this set of regular expression well models date forms, it is not sufficient to properly tag date entities. There are three problems we need to resolve: (1) Lower/Upper case: the orthography of day's and month's names need to be ignored; (2) in Italian, final letters in day's names ended with the vocal "i" are often accented; (3) we can easily see that the expression $dayMonth$ is contained in $dmy1$, $dmy2$ and $dmy3$ while $dayAlone$ is contained in all others regular expressions and this fact generates conflicts when applying REs.
First two issues have been resolved through rapid expedients that the implementing language allows, while for the last problem (avoiding conflicts when applying REs) the policy followed is the longest match. Figure 4.6 shows that the entity is detected through the regular expression $dmy1$ (represented by green arrow) by longest match, correctly discarding other regular expressions $dayAlone$ and $dayMonth$ (represented by red arrows).
Recalling the press review took as example in chapter 3, the Date tagger module detect three *Date* entities:

1. "Martedi 24 Luglio" inside row 11;
2. "Giovedi 26 Luglio" inside row 17;
3. "Lunedi 21 maggio" inside row 22.

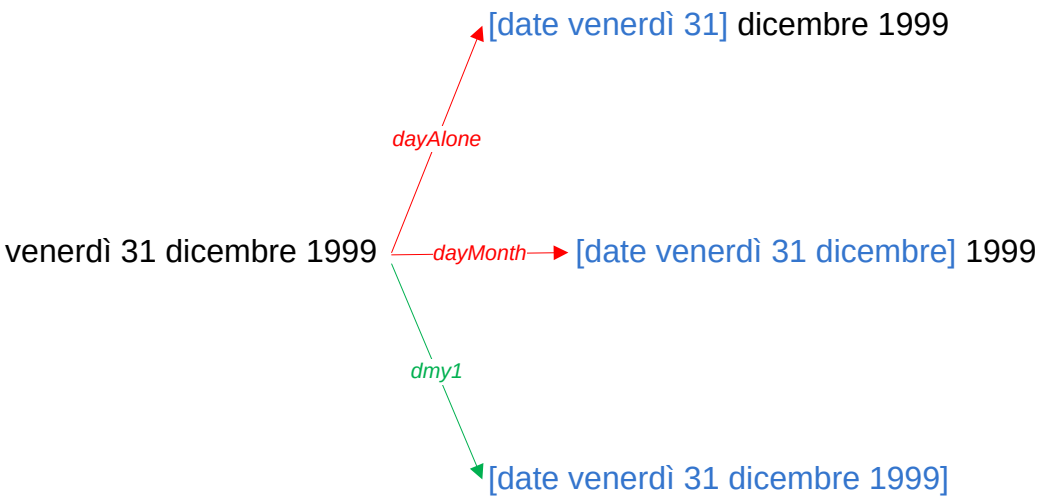

Figure 4.6: Applying REs with longest match criteria

```
10 Anna Calvi fara' tappa in Italia per due date:
11 [date Martedi 24 Luglio]
12 Roma  Parco di San Sebastiano
13 Roma Vintage
14 Via di Porta San Sebastiano 2 (P.le Numa Pompilio), 00187 Roma
15 Biglietto: 15,00 euro + d.p.
16 Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
17 [date Giovedi 26 Luglio]
18 Bologna - Bolognetti Rocks
19 Rocker Festival
20 Vicolo Bolognetti, 40125 Bologna
21 Biglietto: 10,00 euro + d.p.
22 Prevendite attive da [date Lunedi' 14 Maggio] sui circuiti vivaticket.it,
23 ticketone.it. I singoli Blackout, Desire, Jezebel, Suzanne and I sono
```

4.2.2 Range tagger

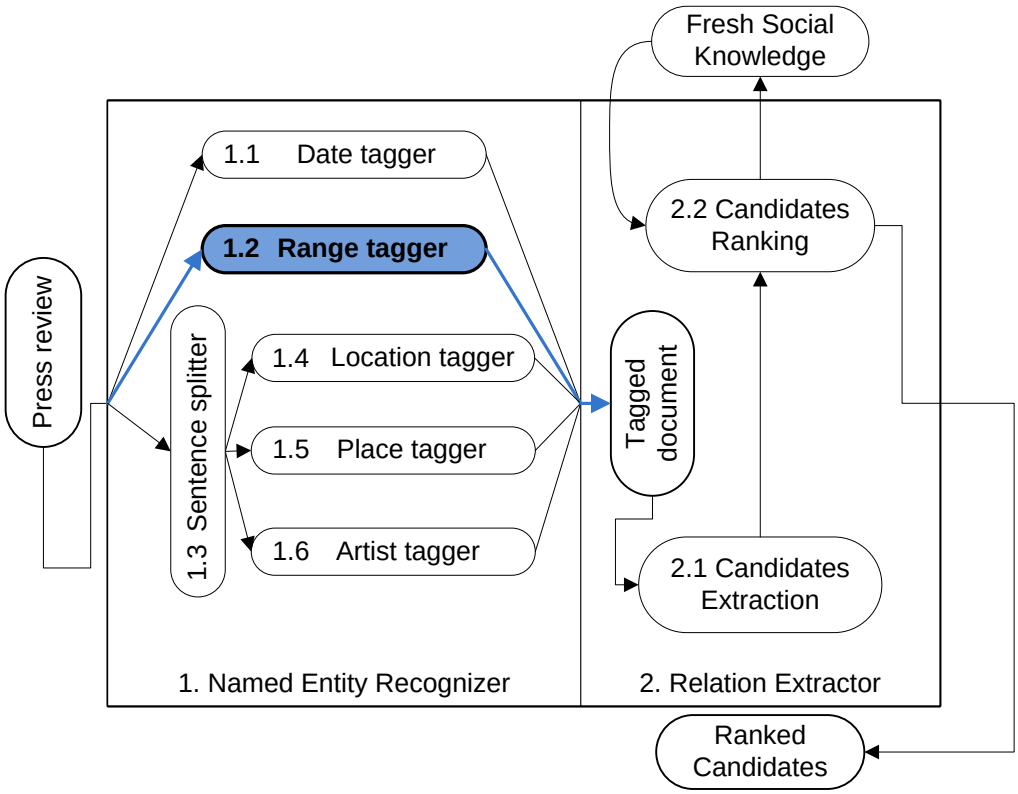

Figure 4.7: Range tagger module

The Range tagger module is committed to tag entities of the class *Range*. As this class of entities fully relies on the Data class, the regular expression already defined for this latter have been used to define the set of REs correspondent to this class.
Date's regular expression have been mapped as:

$DATE$ = dmy1|dmy2|dmy3|dayMonth|dayAlone,

and the set of regular expressions corresponding to the Range class is composed by the following elements:

$range1$ = ((dal|da)[ ]?(dayNumber)(fino)?[ ]?al[ ]? $DATE$);
$range2$ = (((dayNumber)[-,e]+)+$DATE$);
$range3$ = ($DATE$[-]+$DATE$);

$range4 = ((\text{dal}|\text{da})[\ ]?DATE(\ \text{fino})?[\ ]?\text{al}[\ ]?DATE)$.

In our example press reviews, this module correctly does not detect any range.

4.2.3 Sentence splitter

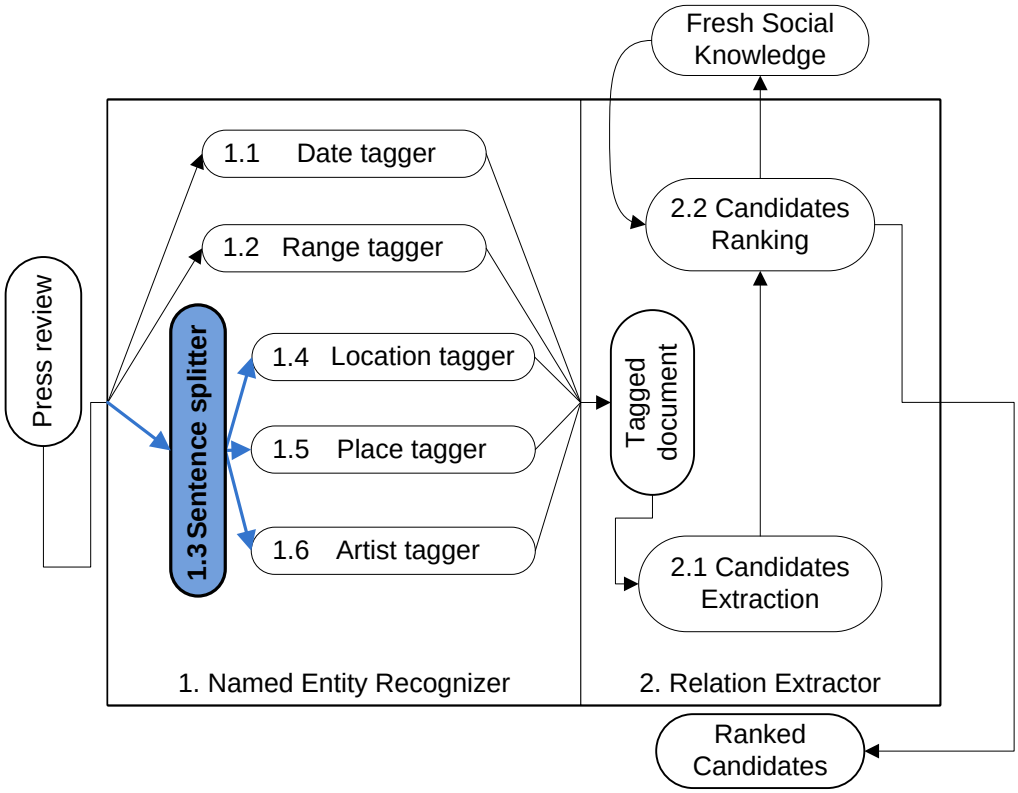

Figure 4.8: Sentence splitter module

The sentence splitter module removes the punctuation and splits the input document in unigrams, bigrams and trigrams in order to facilitate the searches performed by modules 1.4, 1.5 and 1.6.
For example, the sentence "L'omonimo album di debutto l'ha resa celebre in ogni dove coronandola "la nuova musa made in UK"." will be splitted in:

$unigrams$ = {L, omonimo, album, di, debutto, l, ha, resa, celebre, in, ogni, dove, coronandola, la, nuova, musa, made, in, UK};
$bigrams$ = {L omonimo, omonimo album, album di, di debutto, debutto l,

l ha, ha resa, resa celebre, celebre in, in ogni, ogni dove, dove coronandola, coronandola la, la nuova, nuova musa, musa made, made in, in UK };
trigrams = {L omonimo album, omonimo album di, album di debutto, di debutto l, debutto l ha, l ha resa, ha resa celebre, resa celebre in, celebre in ogni, in ogni dove, ogni dove coronandola, dove coronandola la, coronandola la nuova, la nuova musa, nuova musa made, musa made in, made in UK}

In modules that rely on the sentence splitter, text segments generated by this module are compared to the respective dictionaries and if a text segment is found in one of them, the correspondent module recognize the text segment as an entity of the class it is charged to tag.
However, dictionaries often contains name mentions formed by 4 up to 8 words; this can be a problem since the splitting of the input document is computed from 1 up to 3 words. This problem is resolved by splitting the name mentions which exceeds 3 words in trigrams and adding the resulting trigrams as a new entry of the correspondent dictionary. How to recognize an entity with more than 3 words will be a task performed by the next modules.

4.2.4 Location tagger

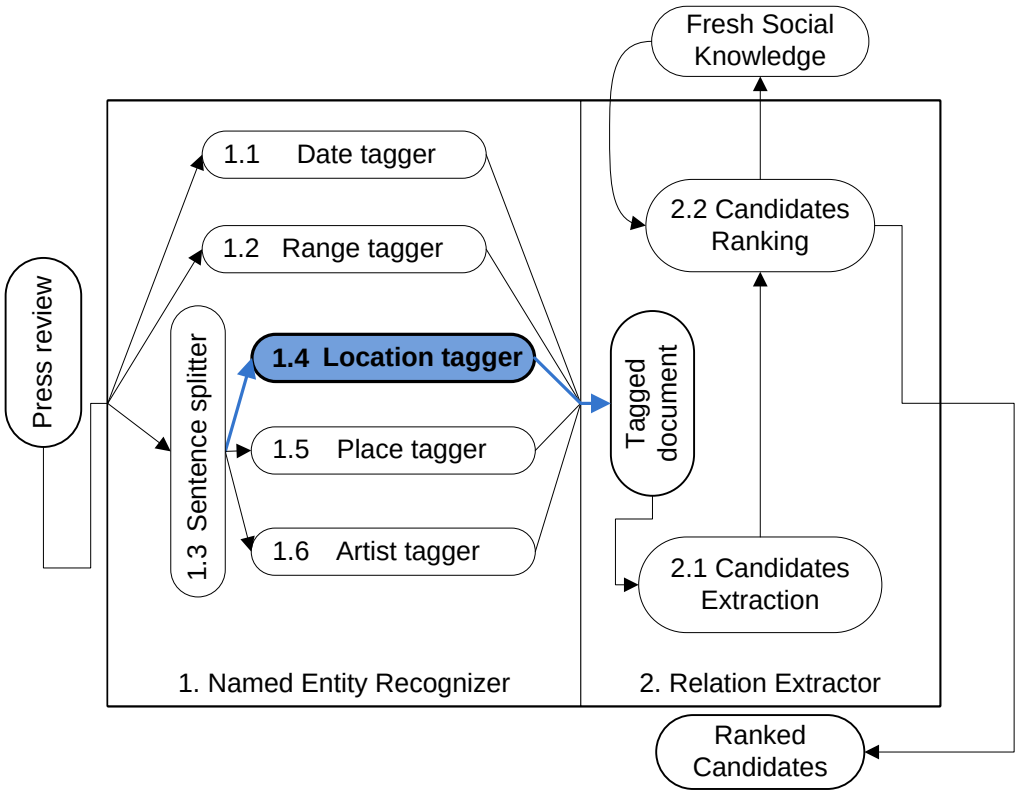

Figure 4.9: Location tagger module

The location tagger module is entrusted to tag entities of the class *Location*. Entities of this type, in our press reviews, are generally name mentions of cities, districts and municipalities in Italy. A list of those kind of entities can be constructed from Wikipedia, which contains a single page for each municipality in Italy.
Figure 4.10 shows the summary of the cities of Venice and Milan founded in the correspondent Wikipedia pages. Starting from this concept, a dictionary containing information about all the italian municipalities has been created in *csv* format. Here's a sample of the dictionary:

```
26063;Treviso;Preganziol;22,89;17025;743,77
26064;Treviso;Quinto di Treviso;18,97;9846;519,03
26065;Treviso;Refrontolo;13,11;1863;142,11
26066;Treviso;Resana;24,97;9351;374,49
26067;Treviso;Revine Lago;18,66;2263;121,28
26068;Treviso;Riese Pio X;30,74;10976;357,06
```

Figure 4.10: Two famous italian municipalities

```
26069;Treviso;Roncade;61,98;14148;228,27
26070;Treviso;Salgareda;27,2;6692;246,03
26071;Treviso;San Biagio di Callalta;48,25;13279;275,21
26072;Treviso;San Fior;17,77;6940;390,55
26073;Treviso;San Pietro di Feletto;19,45;5416;278,46
26074;Treviso;San Polo di Piave;20,94;5021;239,78
26076;Treviso;San Vendemiano;18,39;10091;548,72
26077;Treviso;San Zenone degli Ezzelini;19,97;7459;373,51
26075;Treviso;Santa Lucia di Piave;19,91;9032;453,64
26078;Treviso;Sarmede;17,94;3202;178,48
26079;Treviso;Segusino;18,14;1974;108,82
```

where columns respectively represents municipality's:

- code;
- district;
- name;
- territory in km^2;
- population;

- density of population in $per./km^2$.

The total number of municipalities founded in Italy was 8092 over 110 districts.
To find entities of class *Location*, this module receives in input the output of the sencente splitter module, i.e. unigrams, bigrams and trigrams derived from the sentences. For each one of them a search by exact match is performed in the dictionary. Like the Date tagger module, if a conflict occurs, trigrams are rewarded respect to bigrams and unigrams, while bigrams are rewarder respect to unigrams. The 97% of municipalities is composed by three words, while the remaining 3% is formed from 4 up to 8 words. For this small percentage, name mentions in the dictionary are subdivided in trigrams, and the module checks if one or more trigrams intersects in the input document; if that is the case, it unifies them, fully recognizing the municipality name.
In our example press review, five *Location* entities were found:

1. "Roma", two times inside rows 11, 12 and 13;

2. "Bologna" inside row 17 and 19.

```
10 Martedi 24 Luglio
11 [loc Roma]  Parco di San Sebastiano
12 [loc Roma] Vintage
13 Via di Porta San Sebastiano 2 (P.le Numa Pompilio), 00187 [loc Roma]
14 Biglietto: 15,00 euro + d.p.
15 Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
16 Giovedi 26 Luglio
17 [loc Bologna] - Bolognetti Rocks
18 Rocker Festival
19 Vicolo Bolognetti, 40125 [loc Bologna]
20 Biglietto: 10,00 euro + d.p.
```

4.2.5 Place tagger

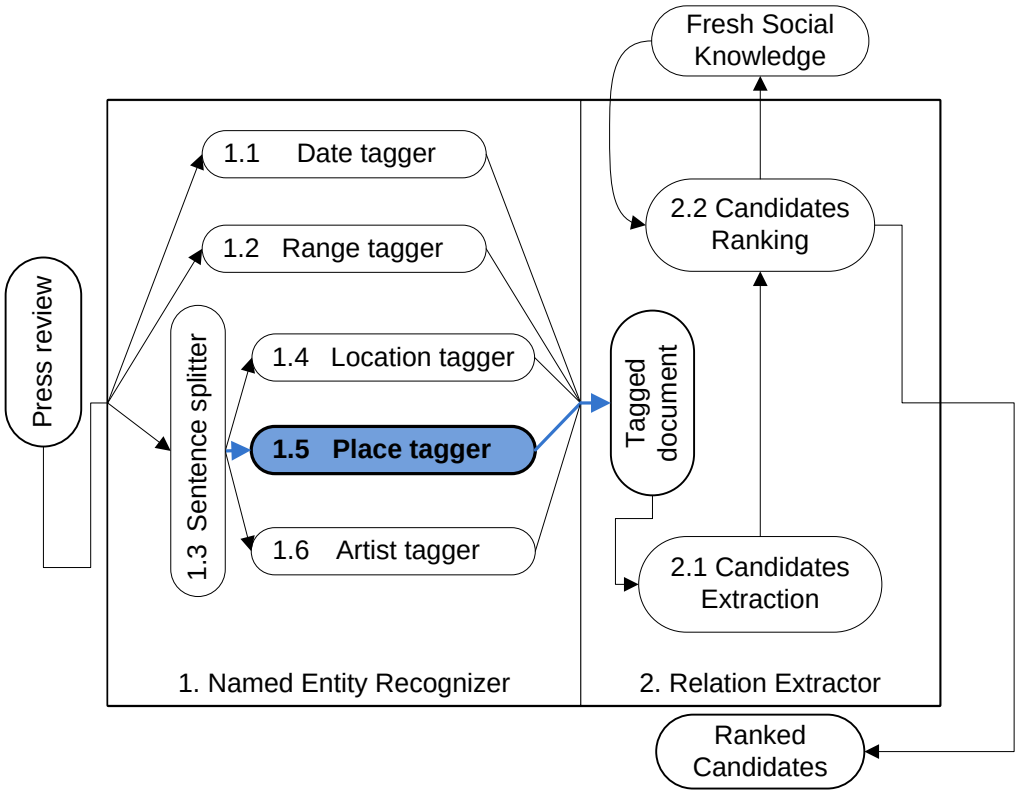

Figure 4.11: Place tagger module

The place tagger module tags entities of the class *Place* present in the unstructured document. Usually, this class of entities includes mentions of locals affiliate with the news agency that advertises their events and writes the correspondent press review.
2night maintains a relational table about locals that uses its services, therefore such table has been used to construct a complete dictionary of places. Here's a sample of the dictionary constructed:

```
"piper caf";http://2night.it/piper-caf-melendugno.html;melendugno;discobar
agor;http://2night.it/agor-cusano-milanino.html;"cusano milanino";circolo
"siddharta caf";http://2night.it/siddharta-caf-milano.html;milano;"cocktail bar"
yguana;http://2night.it/yguana-milano.html;milano;"cocktail bar"
"cape town";http://2night.it/cape-town-milano.html;milano;"cocktail bar"
chakra;http://2night.it/chakra-milano.html;milano;"cocktail bar"
"slice cafe";http://2night.it/slice-cafe-milano.html;milano;"cocktail bar"
"impero caf";http://2night.it/impero-caf-milano.html;milano;"cocktail bar"
delano;http://2night.it/delano-milano.html;milano;"cocktail bar"
```

```
"modus vivendi";http://2night.it/modus-vivendi-milano.html;milano;"cocktail bar"
funhouse;http://2night.it/funhouse-milano.html;milano;"cocktail bar"
skipintro;http://2night.it/skipintro-milano.html;milano;"cocktail bar"
"pura vida";http://2night.it/pura-vida-milano.html;milano;"cocktail bar"
```

The first field corresponds to the name mention of the local, second is the 2night's link for the web page and third field is the category of the local. Currently, the places dictionary contains 16455 locals. Figure 4.12 shows some locals affiliate with 2night.

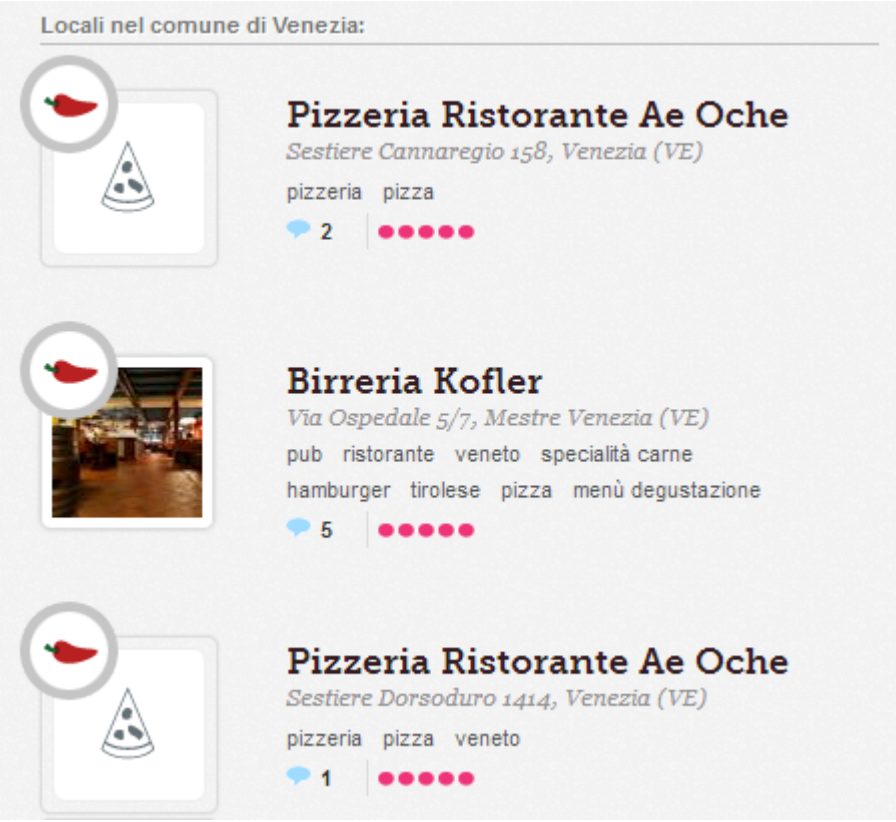

Figure 4.12: Some locals affiliate with 2night

Like the approach took for the Location tagger module, this module takes unigrams, bigrams and trigrams from the output of the sentence splitter module, and a search for exact match is performed for each text segment. More important is given to trigram matches respect to bigram matches which are promoted respect to unigram matches. In our example press review, five *Place* entities were found:

1. "Italia" inside row 10;

2. “Roma Vintage” inside row 13;

3. “Numa” inside row 14;

4. “Vicolo Bolognetti” inside row 20;

5. “Blackout” inside row 23.

```
estivi che la vedranno ospite di numerosi ed importanti festival,
Anna Calvi fara' tappa in [place Italia] per due date:
Martedi 24 Luglio
Roma  Parco di San Sebastiano
[place Roma Vintage]
Via di Porta San Sebastiano 2 (P.le [place Numa] Pompilio), 00187 Roma
Biglietto: 15,00 euro + d.p.
Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
Giovedi 26 Luglio
Bologna - Bolognetti Rocks
Rocker Festival
[place Vicolo Bolognetti], 40125 Bologna
Biglietto: 10,00 euro + d.p.
Prevendite attive da Lunedi' 14 Maggio sui circuiti vivaticket.it,
ticketone.it. I singoli [place Blackout], Desire, Jezebel, Suzanne and I sono
stati accolti con ottime critiche e hanno  riscosso fin da subito
```

4.2.6 Artist tagger

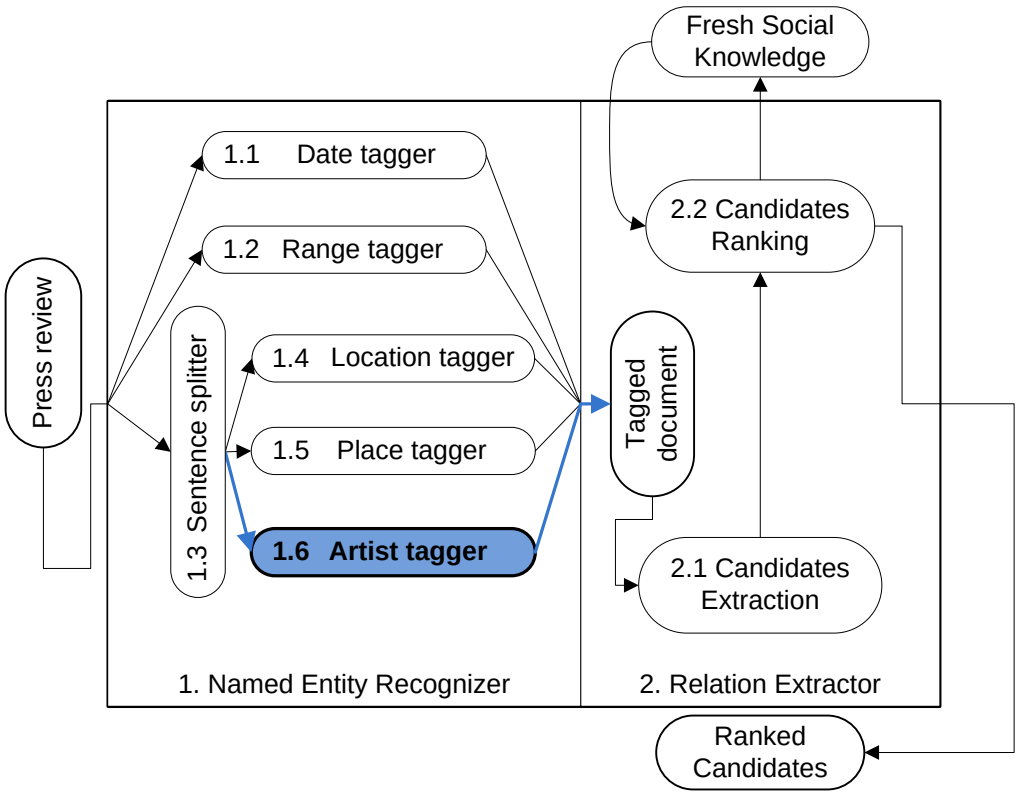

Figure 4.13: Artist tagger module

The artist tagger module searches for entities of the class *Artist*, receiving unigrams, brigrams and trigrams from the sentence splitter module. The dictionary containing artist entities has been constructed from the Italian version of Wikipedia, with an appropriate script that explores all the pages related to the top category "Artisti". Currently, the dictionary contains 45723 entries.
Figure 4.14 shows a fragment of a famous artist from the correspondent italian Wikipedia page. Like modules 1.4 and 1.5, the longest match criteria is adopted when matching text segments. In the press review took as example, eleven Artist entities were found:

1. "Anna Calvi" inside rows 5, 6, 9, 27, 34, 36, 39 and 42;
2. "Nina Simone" and "Scott Walker" inside row 32.

```
4 Un grande successo di pubblico e critica ottenuto grazie alla vincente
5 combinazione di bravura, classe e passione che [art Anna Calvi] riesce ad
```

Figure 4.14: Fragment from Anna Calvi's italian Wikipedia page

```
esprimere con la sua musica e attraverso i live show. [art Anna Calvi] e' una
grande artista, una fuoriclasse. Gia' indaffarata per i prossimi show
estivi che la vedranno ospite di numerosi ed importanti festival,
[art Anna Calvi] fara' tappa in Italia per due date:
Martedi 24 Luglio
Roma  Parco di San Sebastiano
Roma Vintage
Via di Porta San Sebastiano 2 (P.le Numa Pompilio), 00187 Roma
Biglietto: 15,00 euro + d.p.
Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
Giovedi 26 Luglio
Bologna - Bolognetti Rocks
Rocker Festival
Vicolo Bolognetti, 40125 Bologna
Biglietto: 10,00 euro + d.p.
Prevendite attive da Lunedi' 14 Maggio sui circuiti vivaticket.it,
ticketone.it. I singoli Blackout, Desire, Jezebel, Suzanne and I sono
stati accolti con ottime critiche e hanno  riscosso fin da subito
successo e curiosita'. Tutti racchiusi nellalbum di debutto, uscito
per la Domino Records, sono accompagnati da video ricchi di charme
e pathos. Le cover di ogni singolo sono state realizzate con foto e
immagini create da [art Anna Calvi] e Emma Nathan. Anna Calvi: per essere
un album di debutto e' un capolavoro, una magistrale scomposizione
di desiderio e amore. Si sviluppa sulla straordinaria chitarra di Anna
e sulla sua potente e ammaliante voce; e' un album indimenticabile e
appassionante. Influenzata dalle vocalita' di artisti diversi come
```

```
[art Nina Simone], Maria Callas e [art Scott Walker], dalle chitarre di Django
Rheinhard e Robert Johnson, dal classico romanticismo di Ravel e
Debussy, [art Anna Calvi] anche se ispirata da musicisti di un lontano passato,
ha un sound totalmente attuale ma  soprattutto originale. Complici lo
sguardo ipnotico e una bellezza sensuale, [art Anna Calvi] ha conquistato le
copertine ed intere pagine delle migliori riviste e magazine francesi,
tedeschi ed Italiani.
Per i suoi concerti [art Anna Calvi] ha ingaggiato una intima band di appoggio
formata da Mally Harpaz alla chitarra, armonium e percussioni, e Daniel
Maiden alla batteria, e i risultati sono spesso straordinariamente epici.
Benvenuti nel magico mondo di  [art Anna Calvi]  un luogo dove  bellezza e
oscurita' complottano e si scontrano tra loro, dove indomite emozioni
```

Finally the tagged document resulting as output of the first phase of the system is presented. Entities colored in blue are dates, green represents locations, orange identifies places and purples reports artists. In this case, ambiguities are resolved by longest match (row 12: Roma Vintage is longer than Roma).

```
Il 2011 e' stato il suo anno.
L'omonimo album di debutto lha resa celebre in ogni dove coronandola la
nuova musa made in UK.
Un grande successo di pubblico e critica ottenuto grazie alla vincente
combinazione di bravura, classe e passione che [art Anna Calvi] riesce ad
esprimere con la sua musica e attraverso i live show. [art Anna Calvi] e' una
grande artista, una fuoriclasse. Gia' indaffarata per i prossimi show
estivi che la vedranno ospite di numerosi ed importanti festival,
[art Anna Calvi] fara' tappa in [place Italia] per due date:
[date Martedi 24 Luglio]
[loc Roma]  Parco di San Sebastiano
[place Roma Vintage]
Via di Porta San Sebastiano 2 (P.le [place Numa] Pompilio), 00187 [loc Roma]
Biglietto: 15,00 euro + d.p.
Prevendite attive sui circuiti greenticket.it, bookingshow.com e ticket.it.
[date Giovedi 26 Luglio]
[loc Bologna] - Bolognetti Rocks
Rocker Festival
[place Vicolo Bolognetti], 40125 [loc Bologna]
Biglietto: 10,00 euro + d.p.
Prevendite attive da [date Lunedi' 14 Maggio] sui circuiti vivaticket.it,
ticketone.it. I singoli [place Blackout], Desire, Jezebel, Suzanne and I sono
stati accolti con ottime critiche e hanno  riscosso fin da subito
successo e curiosita'. Tutti racchiusi nellalbum di debutto, uscito
per la Domino Records, sono accompagnati da video ricchi di charme
e pathos. Le cover di ogni singolo sono state realizzate con foto e
immagini create da [art Anna Calvi] e Emma Nathan. Anna Calvi: per essere
un album di debutto e' un capolavoro, una magistrale scomposizione
di desiderio e amore. Si sviluppa sulla straordinaria chitarra di Anna
```

e sulla sua potente e ammaliante voce; e' un album indimenticabile e appassionante. Influenzata dalle vocalita' di artisti diversi come [art Nina Simone], Maria Callas e [art Scott Walker], dalle chitarre di Django Rheinhard e Robert Johnson, dal classico romanticismo di Ravel e Debussy, [art Anna Calvi] anche se ispirata da musicisti di un lontano passato, ha un sound totalmente attuale ma soprattutto originale. Complici lo sguardo ipnotico e una bellezza sensuale, [art Anna Calvi] ha conquistato le copertine ed intere pagine delle migliori riviste e magazine francesi, tedeschi ed Italiani.
Per i suoi concerti [art Anna Calvi] ha ingaggiato una intima band di appoggio formata da Mally Harpaz alla chitarra, armonium e percussioni, e Daniel Maiden alla batteria, e i risultati sono spesso straordinariamente epici.
Benvenuti nel magico mondo di [art Anna Calvi] un luogo dove bellezza e oscurita' complottano e si scontrano tra loro, dove indomite emozioni conquistano e consumano.

4.3 RE - Relation Extractor

In this section we explain the novel approach addressing the RE task introduced in section 4.1. We describe what's the scope of the module and two subsection describing the two submodules forming the Relation Extractor: the Candidate Extraction module and the Candidate Ranking module.

Current limitations of state-of-the-art approaches studied in chapter 2 are that they only work by sentence level and on the single input document, but we discovered that our relation can span between sentences and even cross documents.
Using Social Web resources, our approach goes beyond sentence-level fashion spanning outside the single input document when inferencing relations between entities. The input of the RE module is the tagged document coming out from the Named Entity Recognizer module, while the output is the list of ranked candidate tuples. This module has been designed for the interaction with an external social knowledge resource that retrieves other documents about the candidate event; those documents are then analyzed and used to rank candidates and output results.

4.3.1 Candidate Extraction

The candidate extraction module receives in input the tagged document from the Named Entity Recognizer module and returns in output the candidate tuples of our relation, ready to be analyzed from following Candidate Ranking module.
At this point, we are able to distinguish the entities present in the input document and their classes. Formally, the output of the Named Entity Recognizer module is a set of classified entities like:

$$E = \{(e_1, c_1), (e_2, c_2), ..., (e_n, c_n)\};$$

where e_i denote an entity and c_i denote its class contained in the set $C = \{Date, Range, Location, Place, Artist\}$. We can split E in five other subsets, each one for one entity class:

$E_{date} = \{(e_i, c_1), i = 0, .., n, c_1 = Date\};$
$E_{range} = \{(e_i, c_2), i = 0, ..n, c_2 = Range\};$
$E_{location} = \{(e_i, c_3), i = 0, .., n, c_3 = Location\};$
$E_{place} = \{(e_i, c_4), i = 0, .., n, c_4 = Place\};$
$E_{artist} = \{(e_i, c_5), i = 0, .., n, c_5 = Artist\};$

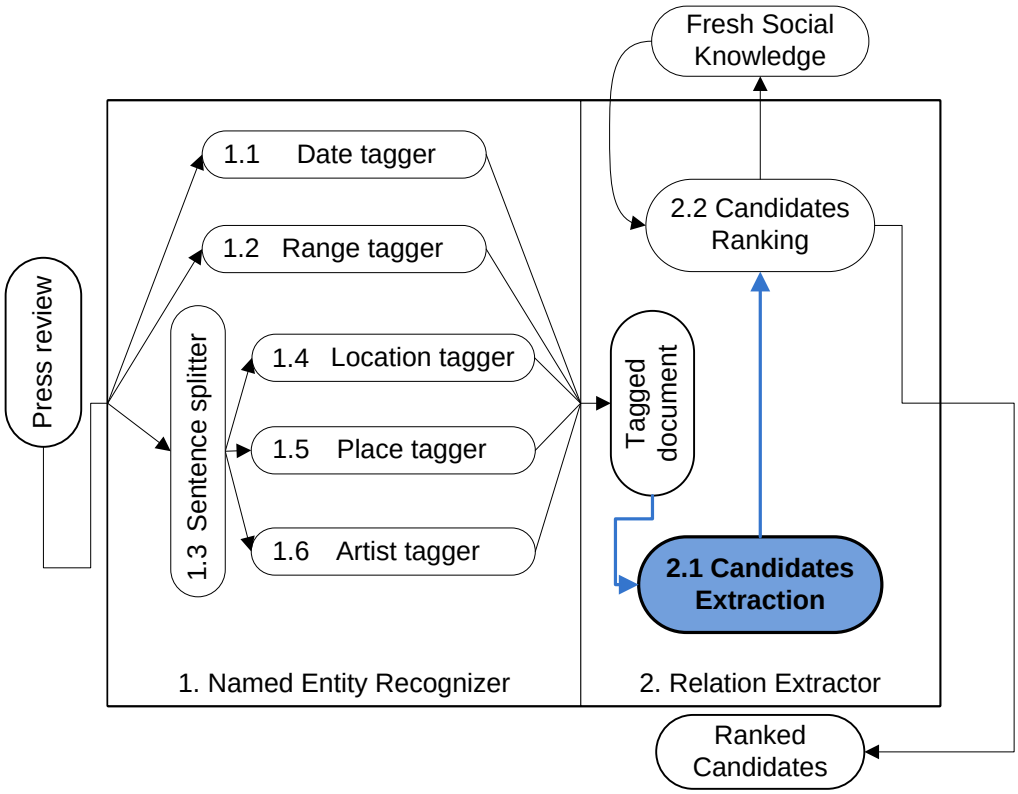

Figure 4.15: Candidate Extraction module

where $E_{date} \cup E_{range} \cup E_{location} \cup E_{place} \cup E_{artist} = E$.
We have also a predefined relation to find:

$$\rho = perform(Date/Range, Location/Place, Artist)$$

The natural way to generate candidate tuples for our relation is to list all the possible combinations of the entities of the constructed sets. Therefore, candidate tuples will be formed by the set:

$$T = \{(dr, lp, a), dr \in (E_{date} \cup E_{range}), lp \in (E_{location} \cup E_{place}), a \in E_{artist}\}. \tag{4.1}$$

Recalling our press review, E is equal to the following set:

$E = \{$(Martedi 24 Luglio, *Date*), (Giovedi 26 Luglio, *Date*), (Lunedi 14 Maggio, *Date*), (Roma, *Location*), (Bologna, *Location*), (Italia, *Place*), (Roma Vintage, *Place*), (Numa, *Place*), (Vicolo Bolognetti, *Place*), (Black-out, *Place*), (Anna Calvi, *Artist*), (Nina Simone, *Artist*), (Scott Walker, *Artist*)$\}$;

the correspondent subsets are equal to:

E_{date} = {(Martedi 24 Luglio, *Date*), (Giovedi 26 Luglio, *Date*), (Lunedi 14 Maggio, *Date*)};
$E_{range} = \emptyset$;
$E_{location}$ = {(Roma, *Location*), (Bologna, *Location*)};
E_{place} = {(Italia, *Place*), (Roma Vintage, *Place*), (Numa, *Place*), (Vicolo Bolognetti, *Place*), (Blackout, *Place*)};
E_{artist} = {(Anna Calvi, *Artist*), (Nina Simone, *Artist*), (Scott Walker, *Artist*)}.

Applying equation (4.1), the set of candidate tuples will be equal to:

T = { ((Martedi 24 Luglio, *Date*), (Roma, *Location*), (Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Roma, *Location*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Roma, *Location*),(Scott Walker, *Artist*)),
((Martedi 24 Luglio, *Date*),(Bologna, *Location*),(Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Bologna, *Location*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Bologna, *Location*),(Scott Walker, *Artist*)),
((Martedi 24 Luglio, *Date*),(Italia, *Place*),(Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Italia, *Place*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Italia, *Place*),(Scott Walker, *Artist*)),
((Martedi 24 Luglio, *Date*),(Roma Vintage, *Place*),(Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Roma Vintage, *Place*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Roma Vintage, *Place*),(Scott Walker, *Artist*)),
((Martedi 24 Luglio, *Date*),(Numa, *Place*),(Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Numa, *Place*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Numa, *Place*),(Scott Walker, *Artist*)),
((Martedi 24 Luglio, *Date*),(Vicolo Bolognetti, *Place*),(Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Vicolo Bolognetti, *Place*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Vicolo Bolognetti, *Place*),(Scott Walker, *Artist*)),
((Martedi 24 Luglio, *Date*),(Blackout, *Place*),(Anna Calvi, *Artist*)),
((Martedi 24 Luglio, *Date*),(Blackout, *Place*),(Nina Simone, *Artist*)),
((Martedi 24 Luglio, *Date*),(Blackout, *Place*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*), (Roma, *Location*), (Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Roma, *Location*),(Nina Simone, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Roma, *Location*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Bologna, *Location*),(Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Bologna, *Location*),(Nina Simone, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Bologna, *Location*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Italia, *Place*),(Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Italia, *Place*),(Nina Simone, *Artist*)),

((Giovedi 26 Luglio, *Date*),(Italia, *Place*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Roma Vintage, *Place*),(Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Roma Vintage, *Place*),(Nina Simone, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Roma Vintage, *Place*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Numa, *Place*),(Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Numa, *Place*),(Nina Simone, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Numa, *Place*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Vicolo Bolognetti, *Place*),(Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Vicolo Bolognetti, *Place*),(Nina Simone, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Vicolo Bolognetti, *Place*),(Scott Walker, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Blackout, *Place*),(Anna Calvi, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Blackout, *Place*),(Nina Simone, *Artist*)),
((Giovedi 26 Luglio, *Date*),(Blackout, *Place*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*), (Roma, *Location*), (Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Roma, *Location*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Roma, *Location*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Bologna, *Location*),(Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Bologna, *Location*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Bologna, *Location*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Italia, *Place*),(Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Italia, *Place*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Italia, *Place*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Roma Vintage, *Place*),(Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Roma Vintage, *Place*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Roma Vintage, *Place*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Numa, *Place*),(Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Numa, *Place*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Numa, *Place*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Vicolo Bolognetti, *Place*),(Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Vicolo Bolognetti, *Place*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Vicolo Bolognetti, *Place*),(Scott Walker, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Blackout, *Place*),(Anna Calvi, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Blackout, *Place*),(Nina Simone, *Artist*)),
((Lunedi 14 Maggio, *Date*),(Blackout, *Place*),(Scott Walker, *Artist*)) }.

Going beyond semantics, we form candidate tuples taking into account only the given relationships. In this module, the idea applied is that if some entities are named in the unstructured document, they have the possibility to occur in the relationship that will indicate the entertainment event. At this moment, tuples are ready to be analyzed by the following module called candidate ranking.

4.3.2 Candidate Ranking

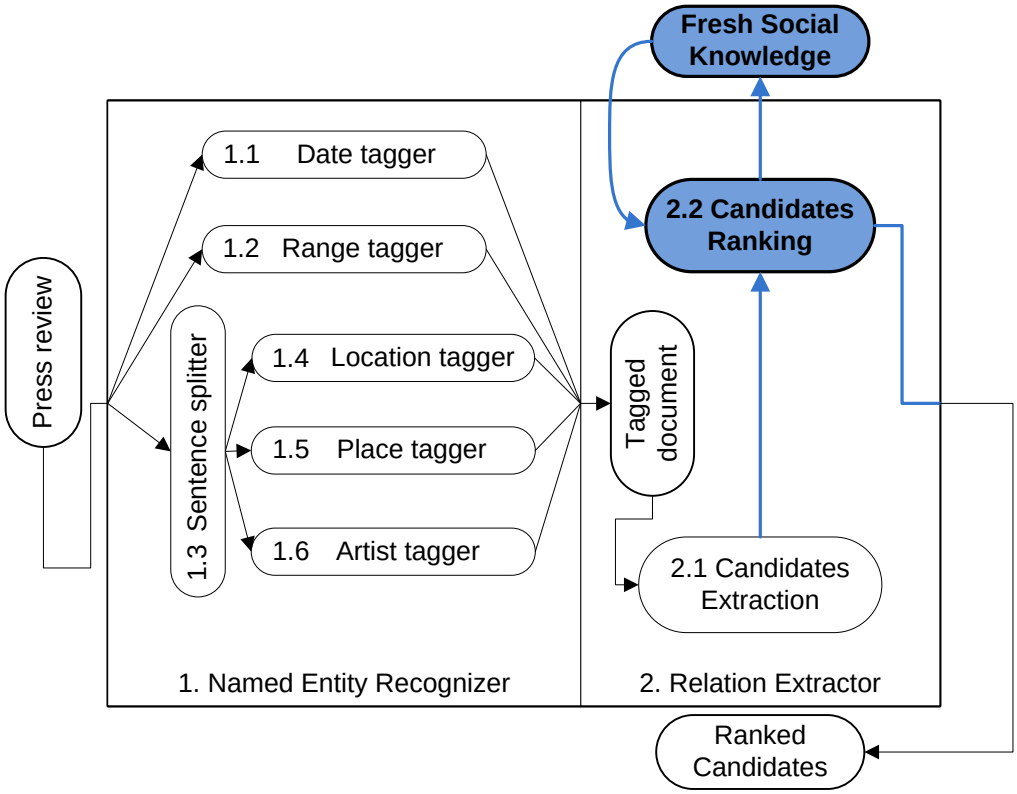

Figure 4.16: Candidate Ranking module

The candidate ranking module receives candidate tuples from the candidates extraction module, then it analyzes and ranks them returning a list of the candidate tuples ranked. Hopefully, top tuple(s) corresponds to entertainment event(s) contained in the unstructured press review. This module relies on an external resource called *Fresh Social Knowledge* (FSK) to rank tuples, and in this subsection we explain how it works.
Given the candidate tuple formed by three entities, we want to analyze other documents for the entertainment event the tuple represents. As explained in the previous section, additional documents concerning an entertainment event are often abundant in the Web. We analyzed various types of sources to make this module interact with the Social Web, and in the following paragraphs we focus on three different external resources.

Encyclopedic knowledge

At first glance, we considered Wikipedia as social knowledge because many researches concerning the IE field works over it; however, Wikipedia represents a static knowledge and it can not be suitable to our task, because its users generally edit a page when a fact has already happened, and we want to extract events that will happen in the future.

Social networks

Another natural candidate for events discovery is the social network Facebook, as it is increasingly widespread and it has strong APIs to work with. Facebook has also a relational table for events; every user can create his own events with simple and rapid clicks. However, it has not been possible to work with this table, because the only field indexed and which can be searched is the name of the event, and there are no standard ways to write it by users (it is just free text without any restriction, there are no rules saying for example that the name of the place, the date or the artist should be in the name column).

Search engine

Learned current limits of Facebook, we decided to use a search engine, *Google*[5]. A search engine generally, given a query, returns documents related which are considered relevant. Starting by a candidate tuple, the query we are going to form is composed by three keywords which correspond to the entities contained in the tuple itself. The candidate ranking module receives the results from the search engine and it scores them. The results are formed by a title (the title of the document) and a snippet (short description of the document). At this moment, only top ten retrieved results are taken into account. Here we present the algorithm developed and used in this module, called *Google Results Scoring* (GRS).

GRS(T)

```
1 R = ∅;
2 ∀ t = (e1, e2, e3) ∈ T:
3     score_t = 0;
4     q_t = t.e1 · t.e2 · t.e3;
```

[5]http://www.google.com

```
5    r_t = first ten results retrieved by searching for q_t;
6    score_t = score_t + score(t, r_t);
7    R = R ∪ (t, score_t);
8  sort(R);
9  return R
```

score(t, r_t)

```
1  s = 0;
2  ∀ r_i ∈ r_t:
3    title_i = r_i.title;
4    snip_i = r_i.snippet;
5    s_t = count(t.e1, title_i) * count(t.e2, title_i) * count(t.e3, title_i);
6    s_s = count(t.e1, snip_i) * count(t.e2, snip_i) * count(t.e3, snip_i);
7    s = s + (10 * s_t + s_s);
8  return s
```

Let's see in details how it works, by explaining the meaning of each step:

1. "R = $\emptyset$"; this step initializes the set of results to the empty set.
2. "$\forall\ t = (e_1, e_2, e_3) \in T$"; a loop starts iterating over the candidate tuples of the set T extracted from the candidates extraction module, the tuples contain three entities named e_1, e_2 and e_3.
3. "$score_t = 0$"; the score for each tuple is initialized to zero.
4. "$q_t = t.e_1 \cdot t.e_2 \cdot t.e_3$"; each entity inside the tuple represents a keyword, and keywords are concatenated forming the query to be searched.
5. "r_t = first ten results retrieved by searching for q_t"; the query is given to the search engine, and first ten results are saved; if less results are found, say $0 <= m < 10$, then m results are taken into account.
6. "$score_t = score_t + score(t, r_t)$"; results are analyzed by a scoring function *score* that takes in input the tuple t and the results of its correspondent query r_t.
7. "$R = R \cup (t, score_t)$"; a couple composed by the tuple and its score is added to the results set R.
8. "$sort(R)$"; the results set R is sorted by descendant order regarding the score of each tuple.
9. "return R" results set is returned.

We describe also the scoring function, *score*, defined to score a tuple given the search results corresponding to its query.

1. "$s = 0$"; the score is initialized to zero.
2. "$\forall\ r_i \in r_t$"; a loop starts iterating over the results given by the query search.
3. "$title_i = r_i.title$"; the title of the i-th result is selected.
4. "$snip_i = r_i.snippet$"; the snippet of the i-th snippet is selected.
5. "$s_t = count(t.e1, title_i) * count(t.e2, title_i) * count(t.e3, title_i)$"; the score of result's title is calculated by the product of the number of times the entities are present in the title.
6. "$s_s = count(t.e1, snip_i) * count(t.e2, snip_i) * count(t.e3, snip_i)$"; the score of result's snippet is calculated by the product of the number of times the entities are present in the title.
7. "$s = s + (10 * s_t + s_s)$"; the score is incremented by ten times the score for the i-th title plus the score for the i-th snippet.
8. "return s"; the resulting score s is returned.

In our example press review, top scored tuples are the following:

(Giovedi 26 Luglio, Bologna, Anna Calvi), 1835.65;
(Martedi 24 Luglio, Roma, Anna Calvi), 1597.89;
(Giovedi 26 Luglio, Roma, Anna Calvi), 290.32;
(Martedi 24 Luglio, Bologna, Anna Calvi), 248.65;
(Lunedi 14 Maggio, Roma, Anna Calvi), 212.11;
(Lunedi 14 Maggio, Bologna, Anna Calvi), 161.14;
(Giovedi 26 Luglio, Bologna, Nina Simone), 129.37;
(Giovedi 26 Luglio, Bologna, Robert Johnson), 126.58;
(Martedi 24 Luglio, Ancona, Anna Calvi), 121.31;
(Martedi 24 Luglio, Roma, Scott Walker), 54.61;

first two tuples have a score much higher respect to others, and they correctly lead to the discovery of the two entertainment events from the press review.

Chapter 5

Experimental results

In this chapter we report results obtained by evaluating the whole system. First, the input set used for evaluation is described, then two different chapters, one for each part of the system, are presented. In evaluating the Relation Extractor module, the performances of the system are compared with two baselines described in the same chapter; finally, results are commented.

In order to produce experimental results, we took a sample of an hundred press reviews, provided by the company editorial office. Those press reviews contain entertainment events taking place in the last four months, so hopefully documents about them are abundant and can be retrieved in the Social Web.

5.1 NER evaluation

To evaluate the Named Entity Recognizer that composes the first part of the system, we manually tagged documents according to the standard form defined in section 4.2.
When an ambiguity was found, the system leave it and the entity involved is counted as a false negative. Manually tagged documents and SEED's tagged documents are compared by a proper script, which counts the number of true positives, false positives and false negatives. Table 5.1 summarizes results with precision, recall and F1 measures.
As we can see, the approach taken presents an high precision together with an average recall. This seems reasonable, because our method works by exact match: if an artist, a place or a location are named through their full name present in the dictionary the correspondent entity will be detected, while if entities are named in other forms they will not be detected. Here's a typical

example:

"Stefano Bollani canterá nella piazza centrale di Padova; Stefano dopo la performance si tratterrá per firmare degli autografi ai suoi fans.";

the output of our Named Entity Recognizer will be:

"**[art Stefano Bollani]** canterá nella piazza centrale di **[loc Padova]**; **Stefano** dopo la performance si tratterrá per firmare degli autografi ai suoi fans.".

Here, the first artist entity is detected, while the second (in black bold) is not detected for the reason we stated above.

	Precision	**Recall**	**F1**
Knowledge & Rule based	95.6	63.6	76.4

Table 5.1: NER evaluation results

5.2 RE evaluation

To evaluate the Relation Extractor module, the same well-known measures used in evaluating the first part have been used over two baselines and our framework. In the following paragraphs, baseline methods are explained and results are reported.

Baseline 1

First baseline approach is the simple one. The idea is that if an artist, a place or a location and a date or a range are named in the same sentence of a press review, then a tuple containing them is formed and returned. If more than one class of entity is found in a sentence, nothing is returned. Here's examples of a positive valid instance, a negative valid instance and an invalid instance for this strategy.

(1) **[art Sven Vath]** si esibirá **[date domenica 5 agosto]** a **[loc Gallipoli]**.

(2) Dopo l'enorme successo ottenuto a **[loc Bari]** **[art Sven Vath]** suonerá **[date domenica 5 agosto]**.

(3) Dopo l'enorme successo ottenuto a [loc Bari] [art Sven Vath] suonerá [date domenica 5 agosto] a [loc Gallipoli].

The first sentence represents a positive valid instance and contains three entities of the classes Artist, Date and Location, so the tuple that will be created is (Domenica 5 agosto, Gallipoli, Sven Vath). Second sentence is a negative valid instance, because the tuple formed by the three entities (Domenica 5 agosto, Bari, Sven Vath) does not represent the right event. Third tuple is an invalid instance because in the same sentence we have more than one entity of the class Location. Thus, the first instance represents a true positive and the second a false positive, while the last instance is a false negative because the right event (Domenica 5 agosto, Gallipoli, Sven Vath) is not detected.

Baseline 2

This baseline's idea is that if an artist, a place or a location and a date or a range are named more then the others in a press review, this indicates that the event is formed by those ones, and the correspondent tuple is returned. A simple count over the entities found by the Named Entity Recognizer is performed, and tuples formed by entities with a count score higher than the others are returned. Here's an example (following sets contain tuples formed by entities and their count over a press review):

$(Date_{count} \cup Range_{count})$ = {(domenica 5 agosto, 5), (lunedi 14 maggio, 2), (sabato 4 agosto, 1)}
$(Location_{count} \cup Place_{count})$ = {(Gallipoli, 3), (Bari, 3), (Milano, 1)}
$Artist_{count}$ = {(Sven Vath, 10), (Martin Solveig, 2)}

This method will select (domenica 5 agosto, 5) from the first set, (Gallipoli, 3), (Bari, 3) from the second set and (Sven Vath, 10) from the third set, as they all have a count score higher than the other entities of the respective sets. Afterwards, tuples are formed by combining the selected entities, and they will be:

(Domenica 5 agosto, Gallipoli, Sven Vath),
(Domenica 5 agosto, Bari, Sven Vath);

the first one will be marked as a true positive because it indicates the right event, while the second one will be considered as a false positive.

In order to produce valid results, each press review of our sample set has been manually analyzed and tuples indicating valid entertainment events have been manually produced and compared with the results of the two baselines and our framework SEED. The following table shows the precision, recall and F1 measure for each one of the strategies adopted.

	Precision	**Recall**	**F1**
Baseline1	59.1	14.6	23.4
Baseline2	27.4	50.6	35.6
SEED	63.2	79.6	70.5

Table 5.2: RE evaluation results

First baseline presents a precision of almost 60% and a low recall; this is reasonable because the major part of our relations can't be detected working by sentence, as stated in section 4.1. The average precision score is due to the fact that when a sentence forms only one tuple, that tuple probably correspond to the right event.
Second baseline has an average recall and a low precision. This is due to the fact that this method is based on entity counts: this approach returns relations with an entity's count score higher than the others, and in these relations the right events can be present, but together with a lot of false positives.
SEED presents an average precision and an high recall, with a final F-measure around 70%. Even if wrong entities are detected by the Named Entity Recognizer, this method is still able to understand the right relation, distinguishing also right entities between the wrong ones.

Chapter 6

Conclusions

In this thesis we faced a problem proposed by a company affiliate with our university. The scenario involves detecting new events from press reviews, a task which is currently hand made by journalists of the company's editorial office. Starting from verbose, ambiguous and long press reviews, journalists have to read, analyze and understand them in order to generate new entertainment events that will be advertised by the company. This process is often prolix and it can be considered as a concrete loss of working hours.
The problem outlined involves *Information Extraction* (IE), a discipline aiming the extraction of structured information from unstructured sources, and two subtasks related to it: (1) *Named Entity Recognition* (NER) whose function is to recognize entities from unstructured text and (2) *Relation Extraction* (RE) which tries to understand semantic relations between entities. More specifically, in our problem the NER task aims to find entities of the classes *Date*, *Range* (entities formed by continuous dates), *Location* (municipalities in Italy), *Place* (locals affiliate with the company) and *Artist*, while the RE task aims to find 3-ary relations forming entertainment events under predefined forms, which can be:

- $(Data, Location, Artist)$,
- $(Data, Place, Artist)$,
- $(Range, Location, Artist)$,
- $(Range, Place, Artist)$;

those tuples contain classes of entities and represents what an entertainment event is, indicating that an artist is performing in a certain place or location on a precise date or set of dates.
In this dissertation as first we described the background and analyzed the

state of the art solutions addressing our problems, distinguishing in two different sections the related work concerning NER and RE respectively. Afterwards, we analyzed the scenario we encountered and motivations which lead us to define a novel strategy in RE, different from the previous well-known solutions, which exploits the potential of today's Social Web in inferring entertainment events. Basically, the idea is to start from the input press review, retrieve more documents concerning the candidate events hidden in it, analyze those new documents and finally detect right entertainment events by scoring candidates. A framework called *Social Entertainment Events Detection* (SEED) has been designed in application of our new method, and SEED's implementation choices and submodules have been fully documented in the respective sections.

To verify our innovative strategy, well-known measures have been evaluated over a sample of an hundred press reviews kindly provided by the company. In addition to our solutions, two RE baselines have been designed end evaluated with the same measures, (1) one simply reporting events whose entities are placed in the same sentence, (2) one based on an entity count score to form events. Both these baselines work regarding only the single input document, they do not regard other social resources and their performance are quite lower than SEED. Indeed, the strength of our framework is represented by the candidate ranking module, which works together with an external module of *Fresh Social Knowledge* (FSK) which is viewed like a plugin used for scoring tuples: it can be a search engine like we did or it can be a social network as long as it can rank our candidate tuples.

Finally, SEED's results are encouraging and confirm our approach: our relation between entities can span over the single sentence and even cross documents.

Chapter 7

Future works

In this last chapter we want to describe some future works inspired from results obtained by our novel method, some of these will be implemented and used to improve the performance of our framework. Each of the following paragraphs describe a possible future work.

New social knowledge bases

In our implementation, SEED's candidate extraction module works together with a famous search engine which is seen as a social knowledge, and results outlined widely rely on this external resource. Obviously, this social knowledge can be substituted with others present in the Social Web. Twitter, for example, is a natural candidate for this: famous artists have often a personal account and they (or their public relations teams) frequently write *tweets* explaining where and when they will perform to attract more audience. Another candidate social resource is Facebook: even if its event table is not suitable to our task as explained in section 4.3.2, artists often have a fan page and some fans are used to write in it some comments about the artist's performance they will attend. Moreover, we can use several external resources instead of using only one of them and detecting events by comparing the scored tuples of each social resource.

Improving NER phase

Currently, the relation extraction module fully relies on the named entity recognizer. As we saw in section 5.1, an error is introduced when the named entity recognizer module tries to detect entities of various classes, and this error is propagated to the relation extraction module. A possible future work can be to improve the performance of the named entity recognizer by

using other techniques seen in section 2.2. Moreover, the relation extraction module can be used to inform the named entity recognizer module of the false entities it extracts, as it is able to distinguish false positives wrongly extracted (i.e. tuples containing false positives are penalized respect to others in the ranking phase); this will lead to an higher precision score when performing the NER task.

Learning the ranking function

The candidate ranking module's ranking function described in section 4.3.2 relies on some parameters which have been experimentally tuned, but in the future it can be possible to learn the optimal ranking function by analyzing precision and recall scores with machine learning techniques.

Auto disambiguation

As described in section 4, our named entity recognizer leave an ambiguity when it is encountered, and the ambiguous entity is counted as a false negative in the evaluation phases. However, some valid techniques described in [20] and [21] are available and suitable for this problem. Resolve ambiguities will decrease the number of false negatives, and consequently the recall score of our NER approach will increase.

Real time modifies of Wikipedia via Twitter

Wikipedia contains static pages regarding various entities, and its editor users are supposed both to write new pages concerning new entities and to update current pages. It happens very often that pages are modified after some facts (for example, the page referring to all the Presidents of the United States needs to be updated whenever a new president is elected) that can be traced regarding trends over the social network Twitter. In this sense, our work can be used to announce which artist pages in Wikipedia needs to be updated; for example if an artist does a concert, this entertainment event probably will be reported in his Wikipedia page. A challenging future work is to suggest which pages of Wikipedia needs to be updated by extracting entities and relations from *tweets* forming generic events.

Bibliography

[1] S. Sarawagi, *"Information Extraction"*, Foundations and trends in databases, 2008.

[2] N. Bach, S. Badaskar, *"A Review of Relation Extraction"*, Literature review for Language and Statistics II, 2007.

[3] M.Kaufman, *"Defense Advanced Research Projects Agency Report"*, Proceedings of the 1st Message Understanding Conference (MUC-1), 1987.

[4] M.Kaufman, *"Defense Advanced Research Projects Agency Report"*, Proceedings of the 3st Message Understanding Conference (MUC-3), 1991.

[5] , M.Kaufman *"Defense Advanced Research Projects Agency Report"*, Proceedings of the 5st Message Understanding Conference (MUC-5), 1993.

[6] , M.Kaufman *"Defense Advanced Research Projects Agency Report"*, Proceedings of the 6st Message Understanding Conference (MUC-6), 1995.

[7] , M.Kaufman *"Defense Advanced Research Projects Agency Report"*, Proceedings of the 7st Message Understanding Conference (MUC-7), 1998.

[8] , A. Mikheev, M. Moens, C. Gover, *"Named Entity Recognition without Gazetteers"*, in Proceedings of EACL, 1999.

[9] F. Ciravegna, *"Adaptive information extraction from text by rule induction and generalisation"*, in Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI2001), 2001.

[10] D. Koller, L. Getoor, N. Friedman, A. Pfeffer, B. Taskar, *"Probabilistic Relational Models"* in Introduction to Statistical Relational Learning, 2007

[11] C. J. van Rijsbergen, *Information Retrieval*, Information Retrieval Group, University of Glasgow, 1979.

[12] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, *"Text classification using string kernels"*, Journal of Machine Learning Research, 2001.

[13] R. C. Bunescu, R. J. Mooney, *"Subsequence kernels for relation extraction"*, Neural Information Processing System, NIPS 2005.

[14] D. Zalenko, C. Aone, A. Richardella, *"Kernel methods for relation extraction"*, Journal of Machine Learning Research, 2003.

[15] R. McDonald, F. Pereira, S. Kulick, S. Winters, Y. Jin, P. White, *"Simple algorithms for complex relation extraction with applications to biomedical ie"*, in Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, Ann Arbor, Michigan, 2005.

[16] D. Yarowsky, *"Unsupervised word sense disambiguation rivaling supervised methods"*, in Proceedings of the 33rd conference on Association for Computational Linguistic, NJ USA,1995.

[17] S. Brin, *"Extracting Patterns and Relations from the World Wide Web"*, WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT 1998.

[18] E. Agichtein, L. Gravano, *"Snowball: Extracting relations from large plain-text collections"*, in Proceedings of the Fifth ACM International Conference on Digital Libraries, 2000.

[19] M. Banko, M. J. Carafella, S. Soderland, M. Broadhead, O.Etzioni, *"Open information extraction from the web"*,
in Proceedings of the 20th International Joint Conference on Artificial Intelligence, 2007.

[20] D. Milne, I. H. Witten, *"Learning to link with Wikipedia"*, in Proceedings of the 17th ACM CIKM.

[21] X. Han, L. Sun, J. Zhao, *"Collective Entity Linking in Web Text: A Graph-Based Method"*, in Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval.