SEED: A Framework for Extracting Social Events from Press News

Salvatore Orlando
DAIS - Università Ca' Foscari
Venezia, Italy
orlando@unive.it

Francesco Pizzolon
DAIS - Università Ca' Foscari
Venezia, Italy
pizzolon.francesco@gmail.com

Gabriele Tolomei
DAIS - Università Ca' Foscari
Venezia, Italy
gabriele.tolomei@unive.it

ABSTRACT

Everyday people are exchanging a huge amount of data through the Internet. Mostly, such data consist of *unstructured* texts, which often contain references to *structured* information (e.g., person names, contact records, etc.). In this work, we propose a novel solution to discover *social events* from actual press news edited by humans. Concretely, our method is divided in two steps, each one addressing a specific Information Extraction (IE) task: first, we use a technique to automatically recognize four classes of *named-entities* from press news: DATE, LOCATION, PLACE, and ARTIST. Furthermore, we detect social events by extracting *ternary relations* between such entities, also exploiting evidence from external sources (i.e., the Web). Finally, we evaluate both stages of our proposed solution on a real-world dataset. Experimental results highlight the quality of our first-step Named-Entity Recognition (NER) approach, which indeed performs consistently with state-of-the-art solutions. Eventually, we show how to precisely select *true* events from the list of *all* candidate events (i.e., all the ternary relations), which result from our second-step Relation Extraction (RE) method. Indeed, we discover that *true* social events can be detected if enough evidence of those is found in the result list of Web search engines.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*Text analysis*; I.5.4 [**Pattern Recognition**]: Applications—*Text processing*

Keywords

Information extraction; Named-entity recognition; Relation extraction; Social event discovery

1. INTRODUCTION

In the last two decades, a huge amount of data are increasingly become available due to the exponential growth of the *World Wide Web*. Though heterogeneous, the vast majority of these data are *unstructured* texts, which anyway often refer to more *structured* information, such as person names, company names, contact records, etc..

In this paper, we propose a solution to a real problem raised up by a Web company, namely to detect structured information about *social events* from unstructured *press news*. The company's mission is to *spread*, *advertise*, and *recommend* cultural events to people for their leisure time, mostly, yet not only, through the Web. Concretely, it focuses on italian events occurring in several places and dates, performed by several national and international artists.

So far, events are manually recognized by members of the company's editorial office before being published on the company's Web site. In a nutshell, several journalists carefully read and inspect long and ambiguous press news looking for significant information about actual events. It turns out that this process may be prolix and lead to a waste of working hours. Thereby, the (semi-)automatic discovery of events from press news is definitely a challenging task, which in turn may help the company accelerate its whole business.

This is an instance of the more general *Information Extraction* (IE) problem, which refers to the discovery of structured information from unstructured data sources (i.e., typically texts). More precisely, in this work we consider two IE-related tasks: *(i) Named-Entity Recognition* (NER) and *(ii) Relation Extraction* (RE). The former aims to extract and classify *entities* from unstructured text. In our scenario, this turns out to detecting the following classes of entities from press news: DATE, LOCATION (i.e., municipalities in Italy), PLACE (i.e., places affiliated with the company), and ARTIST. The latter tries to identify relations between entities. In our case, relations represent events by means of 3-ary tuples connecting our entity classes in the following way: (ARTIST, LOCATION, DATE) and (ARTIST, PLACE, DATE). These two kinds of tuples well describe entertainment events indicating that a specific artist is performing in a certain place or location on a precise date.

In this work, we introduce *Social Entertainment Event Detection* (SEED), a framework that achieves both the IE tasks. Concerning the NER stage, SEED does not make use of any statistical learning method. In fact, since entities are well-known by the company, they can be extracted simply through *regular expressions* and perfect matching with existing backend database of entities (i.e., *gazetteers*).

Conversely, the main novelty of this work regards the RE task. Usually, solutions to RE limit their scope to an *individual* sentence of the single text document, which the entities have been previously extracted from. However, in our scenario, relations can span over the single sentence and sometimes even across several press news. For instance, it may happen that an artist, a place, and a date – which are named in the same sentence – are not referring to a true entertainment event.

WWW 2013 Companion, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2038-2/13/05.

Instead, SEED infers and disambiguates relations between previously discovered entities by exploiting the so-called "*wisdom of the crowd*", namely by using the potential of the *Social Web*. In particular, SEED extracts actual social events from a set of candidates (i.e., triple of discovered entities). Each candidate event is then *ranked* according to their *relevance* on the Social Web. It turns out that such "relevance" could be obtained from several existing Web sources, e.g., *social networks*, *blogs*, *wikis*, *search engines*, etc.

Here, we propose to ask a Web search engine for each candidate event, and to assign it a *relevance score* that is proportional to the ranking positions of those documents returned as relevant by the search engine. The rationale of this intuition is that very likely true events turn out to retrieve several high-ranked Web search results, thereby they may obtain high relevance scores. On the other hand, events that are formally correct (because referring to 3-ary relations between valid entities) yet unreal have definitely less chance of being assigned with top scores.

The remaining of this paper is organized as follows. In Section 2 we discuss the most important related work concerning both the IE-related tasks addressed. Section 3 describes some useful background concepts about NER and RE. Furthermore, Section 4 introduces SEED, namely our solution to the problem of discovering social events from unstructured press news. To evaluate our framework, in Section 5 we present the experiments we conducted on a real-world collection of press news. Experimental results are showed for SEED and for two baseline approaches used for comparison. Finally, Section 6 summarizes our work and highlights possible future research goals.

2. RELATED WORK

Two main research issues are addressed in this work, both related to *Information Extraction* (IE), which generally aims at finding *structured* information from *unstructured* texts. First, we focus on the problem of recognizing *entities* from raw text (i.e., *Named-Entity Recognition* (NER)) [22]. Second, we discuss how *relationship* between such discovered entities may be extracted (i.e., *Relation Extraction* (RE)) [2]. In the following, we present a selection of related work that are among the most relevant to both issues, separately.

2.1 Named-Entity Recognition (NER)

Knowledge-based methods. Knowledge-based methods are naïve techniques for recognizing named-entities through *gazetteers*, i.e., predefined dictionaries or databases of entities. They usually work in two phases: first, they build a large dictionary for each class of entities to be extracted; then simple heuristics, such as exact match or similarity score, are used to decide whether one or more tokens of the unstructured document is an entity or not.

Mikheev *et al.* [21] have shown that, in that small-scale domain of well-know entities, knowledge-based methods are sufficient to obtain valuable results in NER. On the other hand, these methods lack of flexibility because dictionaries need to be continuously updated, and ambiguity resolution of two entities belonging to different dictionaries is hard [13].

Rule-based methods. Most of the initial systems designed for NER were based on *hand-coded* rules [18, 8]. Similarly to knowledge-based methods, these solutions perform consistently when the task is under a controlled domain. Rule-based systems are typically two-stage: entities are firstly found starting from a collection of rules properly designed by a language expert, and thereby checked and corrected with some policies.

A generic rule is usually expressed in the form $\{Pattern \rightarrow Action\}$, where the pattern captures properties of the examined token (i.e., context, orthography etc.), and the action labels the token found by the pattern.

A typical rule-based NER system is made up of several rules, each of which can be applied for detecting one or more entities. However, sometimes rules may rise conflicts when a text segment, or part of it, is matched by two or more rules. To resolve conflicts, several approaches have been proposed so far: custom policies [8], ordered sets [18] and finite state transducers [12].

If enough labeled examples of a class of entities are available, it is 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. Moreover, algorithms are divided in two categories: *(i) bottom-up*, where specific rules are generalized [7], and *(ii) top-down*, where a generic rule is specialized [23]. Usually, for a specific domain no labeled examples can be found, and rules are hand specified by a language engineer. Anyway, if enough training corpus are available, using rule learning algorithms is faster and lead to better results than coding rules by hand.

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. There exist various decomposition models, which differ one from each other by their granularity working level: *(i) Token-level models*, where the unstructured document is viewed as a sequence of words separated by punctuation (e.g. dots, commas, quotes) forming tokens; *(ii) Segment-level models*, where the input document is divided into chunks (i.e. text segments composed by several words) via natural language parsing techniques; *(iii) Grammar-based models* where, inspired by *Context Free Grammars* used to specify programming languages, a set of production rules defined over terminals is used to express the pattern of an entity.

The following solutions are all *supervised learning* methods, which therefore need hand-labeled training examples. A well-established method is to use *Conditional Random Fields* [15] to model a single joint distribution over the predicted labels of the input tokens. In typical extraction tasks, a chain is suitable to capturing label dependencies and therefore the joint distribution can be treated with Markov random fields [10] to express the interdependence, leading to the conclusion that a label is influenced only by its precedent label. Finally, in the training phase the Likelihood-based method [17] or the Max-margin method [24] can be used, while for the inference phase common methods are the *Mean Average Precision* and the *Expected Features Value* metrics.

2.2 Relation Extraction (RE)

Supervised methods. When enough labeled examples for a specific domain can be found, supervised methods turned out to be suitable solution also for extracting relations between entities. They set the RE problem as a classification problem, whose goal is to identify a function returning all the true relations of an unstructured document. Here, we consider three main approaches.

- *Feature-based methods.* These methods extract from the labeled examples semantic and syntactic features, such as the strings representing the entities, the classes of the entities, the strings occurring before, between and after the two entities and the orthography of entities. These features are in turn used to train a classifier over the labeled examples that will be used to find relations between entities in other documents [14]. The features selected to train the classifier depends on the nature of the problem, and in general it is hard to reach an optimal subset of relevant features.
- *Kernel-based methods.* Given two strings s_1 and s_2, in string kernel methods their similarity score is computed with a kernel function $K(s_1, s_2)$ that relies on the number of substrings that are common between them [16]. The function K is trained over a classifier which models positive and negative examples as context (i.e., words) around the two entities [5] or as parse trees [26], and it is used to detect new instances of the desired relation in new sentences.
- *Shortest path.* Sometimes the predicate between two entities in a sentence is enough to determine both if there is a relation between entities and the nature of the relation [6]. In this approach, sentences are scanned with a dependency parser generating a dependency parser tree, and the kernel function taking as input shortest path between entities is used to compare sentences in order to discover relations.

Among the drawbacks that are common to all the supervised methods are the hardness of making them able to deal with new kinds of relations. Indeed, to be effective these methods need a huge amount of labeled data, which must be provided for each new type of handled relation.

Semi-supervised methods. For some domains, finding labeled data to be used to train classifiers used in supervised methods is too hard. When this happens, a common solution – initially designed for entity disambiguation [25] and successively generalized for the RE task [19] – uses an iterative framework that relies on the output of weak learners as training data for the next iteration, giving by hand a starting seed set. Next two systems fully depend on this.
- *Dual Iterative Pattern Relation Expansion (DIPRE).* The DIPRE system [4] follows the iterative framework and it is able to expand the seed set in order to find other instances of a given binary relation over a collection of documents. Starting from a given seed set containing entities in tuples forming instances of the desired relation, DIPRE finds instances of these tuples in the collection and regard the context of each tuple by saving *prefix*, *middle*, *suffix* (respectively ten characters before, characters between and ten characters after the entities) and *order* (i.e., order in which seed entities are found in the sentence) for each tuple. Next step is the hardest one: given initial tuples with their context, DIPRE generates patterns from them by grouping by *order* and *middle* fields. Afterwards, for each group, the longest common suffix and prefix is computed and, together with fields *middle* and *order*, they form a new pattern, which is used to find new instances of the relation over the collection. New instances will form new patterns and the same operations stated above are performed until a convergence criteria is reached, for example DIPRE can stop when no more occurrences were found in the collection or when 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 [1] has been developed over the same framework and aims to overcome this issue: it gives better performance than DIPRE by using a NER system to tag entities, matching tuples with a similarity function and grouping them with a clustering algorithm. Finally, a confidence score is assigned to each pattern and it is used to measure its quality, giving more importance to instances extracted from patterns with higher score than others.
- *TextRunner.* Unlike DIPRE and Snowball systems, which work on a user-defined relation and require an initial seed to expand instances of the given relation, the "TextRunner" [3] system learns entities, classes, and relations without any human input, and it is composed of three key modules: *(i) Self-Supervised Learner*: given a non-tagged corpus, this component outputs a classifier which tags candidate instances of extraction as reliable or not; *(ii) 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; *(iii) Redundancy-Based Assessor*: it assigns a probability to each retained candidate on the base of a probabilistic model of redundancy in text.

Unsupervised methods. Downey *et al.* [9] propose an *unsupervised learning* technique that is able to extract relations from text without the need of a manually-labeled training corpus. In fact, the authors present a combinatorial "balls-and-urns" model, which learns correct relations on the basis of the *KnowItAll* hypothesis. This states that relations that occur more frequently in distinct sentences in a text corpus are also more likely to be correct. When the text corpus considered is (a part of) the Web, this hypothesis can find many correct relations due to *redundancy*: individual facts/events are often repeated many times and in many different ways on the Web. This is somewhat the same assumption we used for ranking the set of candidates relations as extracted by our RE module (see Section 4).

3. BACKGROUND CONCEPTS

In this section, we formalize both the NER and RE problems, separately. Hereinafter, we refer to $\mathcal{D} = \{D_1, \ldots D_{|\mathcal{D}|}\}$ as a text corpora containing a set of text documents. However, we firstly introduce some basic concepts that are useful to fully understand both the NER and RE issues.

3.1 Classes, Entities, and Mentions

Let us start our discussion from a concrete example and consider the following text document:

```
"Larry Page and Sergey Brin founded Google Inc.
in 1998; in March 1999, the company moved its
offices to Palo Alto, California."
```

Assume that we are interested in finding those substrings within such document referring to a predefined set of *classes*. Similarly to the taxonomy of classes in object-oriented programming, we call a class the *abstract* representation of a concept, which cannot be directly instantiated.

According to this definition, possible examples of classes from the text above are: PERSON, COMPANY, and PLACE. Thus, we refer to an *entity* as a *concrete class*, whereas we call a *mention* each specific *instance* of a concrete class, namely each specific way of expressing an entity within a

document. For example, the mentions "Google Inc." and "Big G" are two possible ways of naming the entity Google, which in turn is a concrete class of the abstract concept of COMPANY.

Informally, the *Named-Entity Recognition* problem aims at discovering the set of entities, as instances of a limited set of classes, from the mentions contained in a particular text document.

As a further step, it could be interesting to derive existing *relations* between entities by inspecting links connecting their related mentions within a given text. Intuitively, a relation may be represented by a *predicate* involving at least two entities (i.e., a *binary relation*). According to this, one possible relation derivable from our sample text is the one explaining "*who* founded *what*". This involves linking the two entities Larry Page and Sergey Brin (both belonging to the class PERSON, and mentioned as "Larry Page" and "Sergey Brin", respectively) to the entity Google (which instead belongs to the class COMPANY, and it is referred to as "Google Inc.").

Thus, given a predicate, the *Relation Extraction* problem consists of linking two (or more) entities as long as they can be related according to that predicate.

3.2 Named-Entity Recognition (NER)

We start focusing on a finite set of *classes* $\mathcal{C} = \{c_1, \ldots, c_{|\mathcal{C}|}\}$. Furthermore, we denote by $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$ the union-set of *all* the possible *entities* associated with all the classes above. Similarly, we define $\mathcal{M} = \{m_1, \ldots, m_{|\mathcal{M}|}\}$ as the union-set of *all* the possible *mentions* referring to all the entities. It turns out that $|\mathcal{C}| \leq |\mathcal{E}| \leq |\mathcal{M}|$.

In addition, let $g : \mathcal{E} \longmapsto \mathcal{C}$ be a function that maps each entity to its corresponding class, and $h : \mathcal{M} \longmapsto \mathcal{E}$ be a function that maps each mention to its corresponding entity.

For the sake of our purposes, we assume each text document $D \in \mathcal{D}$ is represented by a set of mentions $\mathcal{M}_D \subseteq \mathcal{M}$, as follows:

$$\mathcal{M}_D = \{m_i \in \mathcal{M} \mid m_i \text{ is contained in } D\}.$$

The ultimate goal of Named-Entity Recognition is to take as input each document D (actually its representation $\mathcal{M}_D$), and to provide as output a set E as follows:

$$E_D = \bigcup_{m_i \in \mathcal{M}_D} (h(m_i), g(h(m_i))).$$

It turns out that E_D is the set of all the entities along with their related classes, as derivable from the mentions $\mathcal{M}_D$ contained in D:

$$E_D = \{(e_x, c_y) \mid c_y = g(e_x)\}.$$

3.3 Relation Extraction (RE)

From the set E_D defined above, we can easily derive the set $E_{D,c}$ that contains *all and only* those entities of a specific class $c \in \mathcal{C}$:

$$E_{D,c} = \{(e_x, c_y) \in E_D \mid c_y = c\}.$$

Moreover, let $(\hat{c}_1, \ldots, \hat{c}_k)$ be a k-tuple of classes ($\hat{c}_i \in \mathcal{C}$ and $k \geq 2$). We denote by T_D the set of *all* the possible *candidate relations*, defined as follows:

$$T_D = E_{D,\hat{c}_1} \times \ldots \times E_{D,\hat{c}_k}.$$

Thus, the final aim of the Relation Extraction is to detect the set of the *actual* relations $\widetilde{T}_D \subseteq T_D$, such that for each k-tuple of $\widetilde{T}_D$ a given k-ary *predicate* holds.

4. SEED: SOCIAL EVENT DISCOVERY

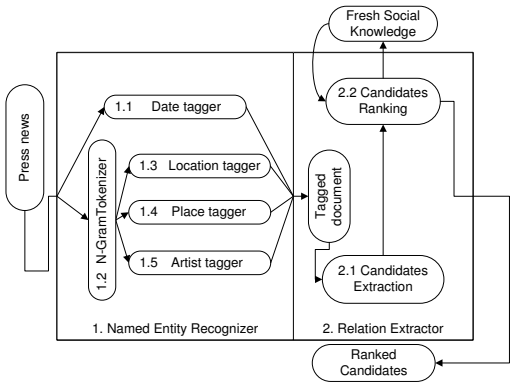

Figure 1: The SEED architecture.

In this section, we describe our proposed framework, called *Social Entertainment Event Detection* (SEED), to discover social events from unstructured press news.

Fig. 1 shows the overall system architecture. The framework is composed of two conceptually-separated modules, each one dedicated to a specific task. For each press news in the whole collection, the *Named-Entity Recognizer* (NER) module generates a *tagged document*. This first module involves five sub-modules (1.1 – 1.5).

Concretely, module 1.2 splits the original text document into n-gram tokens (i.e., blocks of n consecutive words) while modules 1.1, 1.3, 1.4, and 1.5 tag it according to our set of entity classes $\mathcal{C} = \{$DATE, LOCATION, PLACE, ARTIST$\}$. The tagged document output by the NER module has the same structure of the original press news, but it is enriched with *tags* associated with the entities we are looking for.

Furthermore, the tagged document is in turn used as the input of the second module, namely the *Relation Extractor* (RE). There, the *Candidate Extractor* module (2.1) produces *candidate tuples* of relations we aim to extract, namely 3-ary tuples of classes as (ARTIST, LOCATION, DATE) and (ARTIST, PLACE, DATE), which stand for possible events. Then, candidate tuples are processed by the *Candidate Ranking* module (2.2), which ranks tuples by using an external module called *Fresh Social Knowledge* (FSK).

The final output is therefore the list of so scored tuples. Hopefully, highest ranked tuples represent valid instances of our relations, and lead to the discovery of actual entertainment events mentioned in the original press news.

In the following, we describe each module separately.

1) NER Module. By manually inspecting several press news, we conclude the following about the classes of entities we are interested to extract: *(i) Artists* are well-known and basically identified by the mention of the artist/group band; *(ii) Places* can be listed using the company's database of places; *(iii) Locations* can be devised from the Italian ver-

sion of *Wikipedia*[1], which contains an article for each municipality in Italy; *(iv) Dates* have predefined formats.

Moreover, the domain we are operating in is "closed" and well-defined, with no labeled corpus to work with and press news written in Italian. Due to all these aspects, we resort to adopting a *knowledge-based* approach for extracting entities of classes LOCATION, PLACE, and ARTIST. Instead, *rule-based* method appears as the most suitable for detecting entities of class DATE through *regular expressions*.

Apparently, ambiguities may occur when using such simple approaches. In fact, we claim that handling disambiguation is not crucial for our final scope.

Finally, the Named-Entity Recognizer module analyzes the unstructured press news, and collects all the entities it extracts, thereby generating the tagged document through the following sub-modules.

For each press news $D \in \mathcal{D}$ it provides four sets, each one related to a specific class $c_i \in \mathcal{C}$, where $\mathcal{C} = \{\text{DATE}, \text{LOCATION}, \text{PLACE}, \text{ARTIST}\}$, defined as follows:
$E_{\text{DAT}} = \{(e_i, c_d) \mid e_i \in \mathcal{E}, c_d \in \mathcal{C} \wedge c_d = \text{DATE}\}$,
$E_{\text{LOC}} = \{(e_i, c_l) \mid e_i \in \mathcal{E}, c_l \in \mathcal{C} \wedge c_l = \text{LOCATION}\}$,
$E_{\text{PLA}} = \{(e_i, c_p) \mid e_i \in \mathcal{E}, c_p \in \mathcal{C} \wedge c_p = \text{PLACE}\}$,
$E_{\text{ART}} = \{(e_i, c_a) \mid e_i \in \mathcal{E}, c_a \in \mathcal{C} \wedge c_a = \text{ARTIST}\}$,
where the final set $E_D = E_{\text{DAT}} \cup E_{\text{LOC}} \cup E_{\text{PLA}} \cup E_{\text{ART}}$.

Date tagger. This module takes care of tagging tokens representing entities of class DATE. To this end, it uses a set of regular expressions that match with italian, well-known, date formats (e.g., "`DD-MM-YYYY`"). However, this strategy may be not sufficient to properly tag date entities. Any conflict is resolved by returning the token that longest match with the expression. In addition, mentions like "`tomorrow`" or "`next week`" are not yet recognized.

N-Gram tokenizer. This module performs text pre-processing and cleaning (e.g., punctuation removal) on each input press news. Moreover, it splits each original text document into the corresponding set of word n-grams, namely blocks of n consecutive words ($n \in \{1, \ldots, 8\}$) in order to ease the tasks performed by other modules.

Location tagger. This module tags entities of class LOCATION. In our scenario, those entities are generally name mentions of cities, districts, and municipalities in Italy. A list of this kind of entities can be easily derived from Wikipedia, which contains a single article for each existing municipality. Coherently, the total number of italian municipalities in Wikipedia is $8,092$ over 110 districts.

This module takes as input the set of n-grams provided by the *N-Gram tokenizer*, and for each one of them it performs a search for an exact match on the dictionary. If a conflict occurs between two n-grams, the longest match (i.e., the match with the n-gram with the greatest n) is chosen.

Place tagger. The place tagger module tags entities of class PLACE that are present in the original press news document. Usually, this class of entities includes mentions of locals affiliated with the news agency that advertises its events and build the correspondent press news.

The company maintains a relational table of known places. Therefore, such table can be used to build a complete dictionary of places. Like the above module, this takes the word n-grams generated by the *N-Gram tokenizer*, and searches for an exact match on each text segment. Again, conflicts are resolved by returning the longest matches.

Artist tagger. This module extracts entities of class ARTIST, starting from the word n-grams given as input. The dictionary of artist entities is built from the Italian version of Wikipedia by exhaustively scanning the articles related to the corresponding italian top-category (i.e., "*Artisti*"[2]). Currently, the dictionary contains $45,723$ entries. Still, the longest match criteria is used to match n-grams.

2) RE Module. Current solutions to the RE problem usually try to discover relations between entities on a sentence-by-sentence perspective. In other words, they split the input text document into its composing sentences, and identify relations only among entities that occur within each single sentence. However, in our specific context, relations, namely *social events*, may span over multiple sentences and even across several press news.

The main novelty of this work particularly focuses on this task, which we achieve by exploiting the so-called "*wisdom of the crowd*". Specifically, we claim that Social Web sources, such as *social networks*, *blogs*, *wikis*, *search engines*, just to name a few, may be useful to effectively discover events from press news. In this way, we overcome the limitations of state-of-the-art solutions, thereby going beyond the "same-sentence" or even "same-document" boundaries when inferencing relations between entities.

The input of the Relation Extractor module is the tagged document coming out from the Named-Entity Recognizer module, while the output is the list of ranked candidate 3-ary tuples of entities (and related classes) representing the set of detected social events. To reach this goal, it relies on the modules described in the following.

Candidate Extractor. This module takes as input the tagged document from the NER module and generates a set of *candidate* 3-ary tuples, representing *all* the possible social events derivable from the set of discovered entities. Specifically, it focuses on this 3-tuple of classes:

$$(\text{ARTIST}, \text{LOCATION } \textit{or } \text{PLACE}, \text{DATE}).$$

From the 3-tuple above, it is straightforward to derive the set of *candidate events* T_D, as follows:

$$T_D = \{(a, lp, d) \mid a \in E_{\text{ART}}, lp \in (E_{\text{LOC}} \cup E_{\text{PLA}}), d \in E_{\text{DAT}}\}.$$

Candidate Ranking. This is the core module of the whole SEED framework. Its task is to select the final set of 3-ary tuples $\widetilde{T}_D$ from the whole set of candidates T_D, which are supposed to refer to actual events (i.e., $\widetilde{T}_D \subseteq T_D$). True events are those 3-ary tuples of T_D for which the *ternary predicate* "*who* performs *what when*" holds.

To achieve this goal, an external *Fresh Social Knowledge* (FSK) module is used to rank candidate events.

Concretely, given a candidate event $t = (a, lp, d) \in T_D$, the FSK assigns it a *relevance score*. More formally, FSK uses a scoring function $s : T_D \longmapsto \mathbb{R}$ that measures the likelihood of each candidate tuple representing a *true* event. Therefore, from the original set of candidate events T_D, we obtain a new set T'_D whose elements are sorted in descending order according to the scoring function s:

$$T'_D = \{t_i \in T_D \mid s(t_i) \geq s(t_{i+1}) \ \forall i \in \{1, \ldots, |T_D - 1|\}\}.$$

The final set $\widetilde{T}_D$ of the most-likely true events is composed of the top-K elements from T'_D, i.e., $t_1, \ldots, t_K$ and $K \leq |T_D|$.

[1] `http://it.wikipedia.org/`

[2] `http://it.wikipedia.org/wiki/Categoria:Artisti`

Evidence of actual events are abundant in the Social Web (e.g., posts on blogs, events on Facebook, timelines on Twitter, etc.). Thereby, FSK may rely on such external sources to derive relevance scores.

In the following, we examine and discuss three external sources that could be exploited by FSK. However, due to the limits of the first two solutions explained below, in the experimental phase of this work we consider only the third external Web source, namely a real-world Web search engine, as a way of detecting true events.

- *Encyclopedic knowledges.* At first glance, *Wikipedia* could be a suitable source because it has already proven to be useful in handling disambiguation in many IE tasks [20, 11]. However, it represents a *quasi-static* knowledge base that is updated by the community only when a fact or an event has already happened. Conversely, our goal is to extract entertainment events that are yet to come.

- *Social networks.* Other natural choices could be represented by Social Networks like *Facebook*[3], which is increasingly widespread and has well-documented APIs to work with. Facebook also allows its users to create their own events properly "structured" on a relational table with few simple clicks. This apparently makes it the right choice for selecting real events from the set of candidates. However, the only field indexed on such event table – which in turn can be queried – is the name of the event. Unfortunately, since this is just a free-text field, users can freely fill it without having to be compliant with any "standard structure".

- *Web search engines.* Roughly, given a user query, a search engine returns a ranked list of Web documents that are considered relevant to that query. Starting from a candidate event $t = (a, lp, d) \in T_D$, we build up a query q_t, where the search keywords are the concatenation of the *mentions* corresponding to the entities of the tuple itself. The obtained query is thus issued to the search engine, which in turn replies with a *ranked list* of the top-most relevant results, i.e., $R(q_t) = \langle r_1, \ldots, r_l \rangle$.

The candidate ranking module takes $R(q_t)$ and scores each candidate event $t \in T_D$ as follows.

Let $f_{tit}(e, r)$ and $f_{sni}(e, r)$ be the frequency counts of the *mentions* of entity e as measured in the *title* and in the *snippet* of a Web search result $r \in R(q_t)$, respectively. Eventually, the relevance score $s(t)$ of the candidate t can be computed as:

$$s(t) = \frac{\sum_{r \in R(q_t)} \gamma(r) \Big(\alpha \prod_{e \in t} f_{tit}(e, r) + \beta \prod_{e \in t} f_{sni}(e, r) \Big)}{\sum_{\hat{t} \in T_D, r \in R(q_{\hat{t}})} \gamma(r) \Big(\alpha \prod_{e \in \hat{t}} f_{tit}(e, r) + \beta \prod_{e \in \hat{t}} f_{sni}(e, r) \Big)},$$

where $\alpha, \beta \in \mathbb{R}$ are weights introduced for assigning different importance depending on whether the matched entity appears in the title or in the snippet of a Web search result. Furthermore, $\gamma(r)$ is a score assigned to each Web result for taking care of the ranking position of the document where a match with an entity occurs.

We called *Linear* SEED the approach giving the *same* importance to each document retrieved by the search engine, no matter what is its ranking position, i.e., $\gamma(r) = 1$, $\forall r \in R(q_t)$. Furthermore, we named *Non-Linear* SEED the solution where more importance is given to top-retrieved documents. The claim is that retrieved documents that are ranked higher by the search engine should account for more when computing the final relevance score for a candidate event. In other words, having a candidate tuple matching two retrieved documents, we want to give more importance to the match occurred on the highest-ranked document.

Concretely, let $rank(r) \in \{1, \ldots, l\}$ denote the ranking position of the result r on the Web search engine's result list $R(q_t)$. Each result $r \in R(q_t)$ for which a match with an entity e occurs is scaled by a factor that is inversely proportional to its ranking position, namely $\gamma(r) = 1/rank(r)$.

So far, only the top-10 retrieved Web results are taken into account for computing the final relevance scores of candidate events (i.e., $|R(q_t)| = l = 10$).

5. EXPERIMENTS

In this section we describe the experiments we conducted as well as the results we obtained to evaluate our proposed SEED framework.

To assess the validity of our solution, we took a sample of 100 *real* italian press news, provided by the company's editorial office. Each press news was manually-labeled by a member of the editorial office, who discovered a total amount of $1,222$ *entities*, and 198 *events* (i.e., relations between entities). Consistently with the structure of the paper, we present the results separately for each task.

Named-Entity Recognition. To evaluate this task that composes the first part of the system, we consider the set of entities as manually-identified from the press news corpus, and we compared them to the set of entities which were automatically discovered by our NER module.

In particular, for each individual entity class we computed the following indicators: *(i)* the fraction of entities that our automatic NER system labeled correctly among all the entities that it returned as labeled with that class (i.e., *precision*); *(ii)* the fraction of entities that our automatic NER system labeled correctly among all the *true* entities that have been manually-labeled with that class (i.e., *recall*). In addition, we also computed the harmonic mean of the above indicators, which is known as *F-measure*.

We started by evaluating the ability of our NER module in recognizing entities of class DATE, which were extracted using a *rule-based* method, i.e., a set of regular expressions. In Tab. 1 we show the precision, recall, and F-measure scores obtained by our solution.

	DATE
Precision	0.760
Recall	0.811
F-measure	0.785

Table 1: Precision, Recall, and F-measure of the NER module for the class DATE.

Besides, entities of all the other classes have been discovered by a *knowledge-based* method. Thereby, results obtained on those are shown separately. Concretely, in Fig. 2 (a), (b), (c) we present the precision, recall, and F-measure scores for each entity class LOCATION, PLACE, and ARTIST by varying the n-grams used to split each input press news from $n = 1$ up to $n = 8$. Moreover, the plot depicted in Fig. 2 (d) shows the same indicators yet aggregated for all the three classes.

[3] http://www.facebook.com

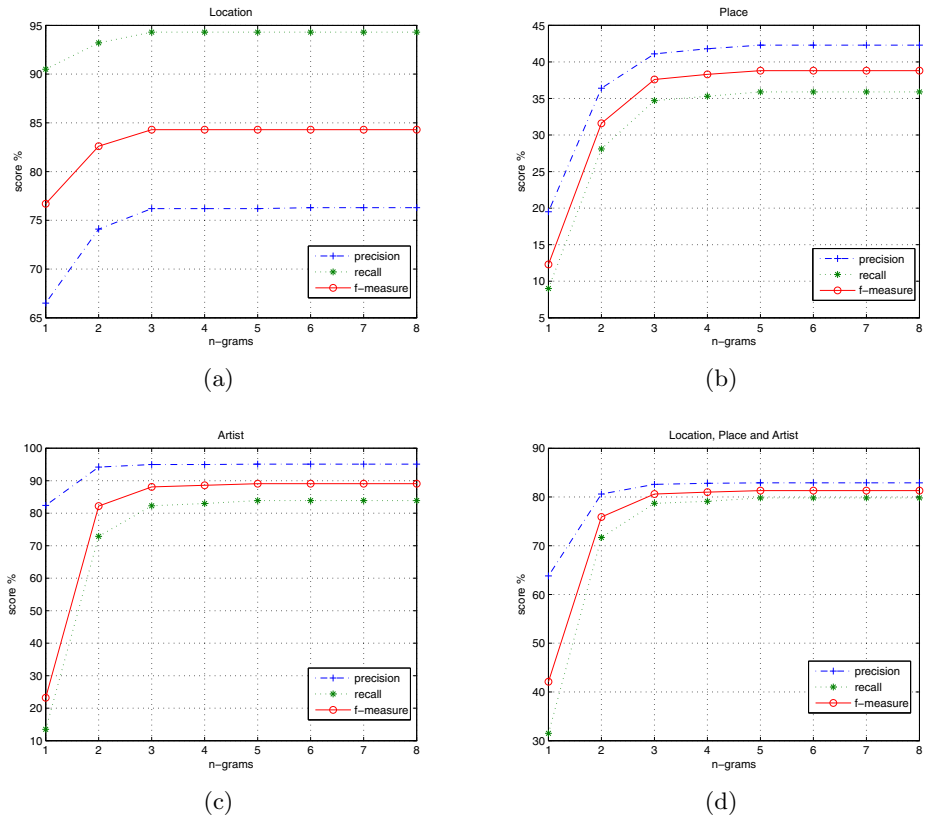

Figure 2: Precision, Recall, and F-measure of the NER module for the classes LOCATION, PLACE, and ARTIST.

Our knowledge-based approach to NER presents a precision which is higher than recall. This seems reasonable, because our method works by finding exact matches: if a mention of an artist, a place (or a location) perfectly matches with an entry of the dictionary of entities then the corresponding entity is detected, otherwise entity is not recognized at all even if a slightly variation on its mention occurs.

Finally, Tab. 2 shows the overall scores of our NER task, taking care of all the four classes considered. The F-measure for the NER module is around 81%, which highlights the quality of our approach, and confirms that for a closed domain like the one where we operated, knowledge-based methods and rule-based methods are truly effective.

	Overall NER task
Precision	0.823
Recall	0.792
F-measure	0.807

Table 2: Overall Precision, Recall, and F-measure of the NER task.

Relation Extraction. In Section 4, we discuss on how the power of the Social Web could be exploited to extract actual events. To this end, we figured out that Web search engines might be effective tools to distinguish between real *versus* fake events. In particular, during this stage we made use of the *Google*[4] search engine to detect actual relations between discovered entities (i.e., events). In addition, the following two *baselines* approaches were also evaluated.

- *Baseline 1*. This is the simplest event discovery method that works as follows. The idea is that if an artist, a place (or a location), and a date are named in the *same* sentence of a press news, then a tuple containing them might refer to a true event and thus it is worth to be returned.
- *Baseline 2*. The rationale for this baseline is to exploit the frequency of entities to extract real events. Indeed, if an artist, a place (or a location), and a date are named more than the others in a press news, this might indicate that very likely the event is formed by such entities, and the correspondent tuple is thereby returned. A simple count over the entities found by the NER module is performed, and tuples composed of entities whose frequency is higher than the others are thereby returned.

As for the NER step, evaluation of any RE method was conducted by comparing the set of manually-labeled events extracted from press news with those automatically provided by our solution. The set of manually-identified events is the *golden set* (i.e., the set of *true* events) that we used for measuring the quality of the two baselines and our RE module by means of precision, recall, and F-measure. To this end, Tab. 3 shows the precision, recall, and F-measure for all the tested RE approaches.

The first baseline presents a precision of almost 60% and a low recall (i.e., about 23%). This is reasonable because the vast majority of our relations cannot be detected by working on a sentence-by-sentence perspective. The quite

[4] http://www.google.it

	Precision	*Recall*	*F-measure*
Baseline 1	0.591	0.146	0.234
Baseline 2	0.274	0.506	0.356
Linear SEED	0.627	**0.798**	0.702
Non-Linear SEED	**0.632**	0.796	**0.705**

Table 3: Overall Precision, Recall, and F-measure of the RE task.

good precision score is due to the fact that sentence that appears in only one tuple likely represents a true event.

The second baseline behaves exactly the opposite: it has a decent recall score (i.e., about 50%) and a low precision (i.e., $\approx$ 27%). This is due to the fact that this method is based on entity counts, and extracts relations for entities whose frequency is higher than the others. It turns out that such relations may contain actual events yet together with a lot of fake ones (i.e., high *false positive* rate).

Finally, to evaluate the RE module of our proposed framework SEED, we considered two aspects both concerning each result r retrieved by the Google search engine: *(i)* the score $\gamma(r)$, and *(ii)* the ratio $m = \alpha/\beta$ between the weights that were given to the title and the snippet of r, respectively. In any case, true events were identified by extracting the top-2 3-tuples from the set of candidates T_D.

In Fig. 3 we show results of *Linear* approach (i.e., $\gamma(r) = 1$), while Fig. 4 depicts the performance of *Non-Linear* method (i.e., $\gamma(r) = 1/rank(r)$), both varying m.

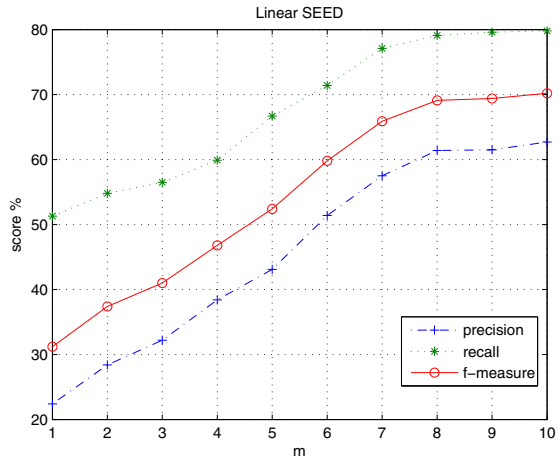

Figure 3: Precision, Recall, and F-measure for *Linear* SEED.

As revealed by these plots, best results were obtained when $\alpha \geq \beta$, namely when more importance were given to entity matches occurring in the document's title instead of those occurring in the snippet.

Linear SEED presents a good precision and a high recall, with a F-measure of about 70.1%. Even if wrong entities were detected by the NER module, this method was still able to understand the right relation, distinguishing also right entities from wrong ones.

Non-Linear SEED had almost the same performance of the Linear SEED, with a F-measure of about 70.5%. Its slightly better performance could result from the higher $\gamma(r)$ scores assigned to high-ranked Web results, which eventually turned out to be the most relevant to extract true events.

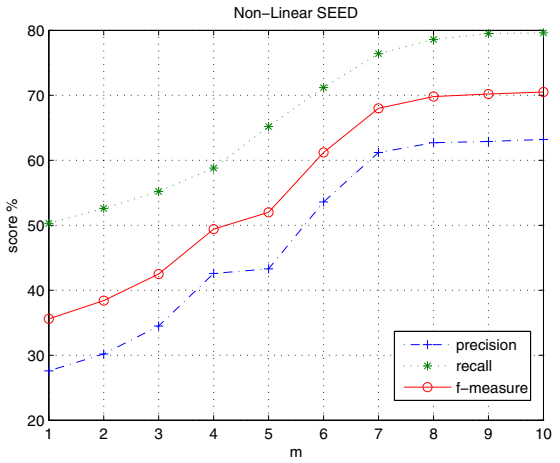

Figure 4: Precision, Recall, and F-measure for *Non-Linear* SEED.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed SEED, a framework to automatically discovery *social events* from a collection of unstructured press news provided by the editorial office of a real-world Web company. Social events were represented by 3-ary relations between specific classes of *entities*. In particular, we focused on four classes, i.e., DATE, LOCATION, PLACE, and ARTIST, and we considered an event any relation of the form (ARTIST, LOCATION, DATE) and (ARTIST, PLACE, DATE).

The SEED framework is made up of two main modules, each one addressing a specific *Information Extraction* (IE) task. From a side, it uses a *(i) Named-Entity Recognizer* (NER) module to extract entities from unstructured texts. On the other hand, it realizes a *(ii) Relation Extractor* (RE) module to discovery relations among previously discovered entities. Since entities were well-defined in the company's domain, no complex, statistical-learning method has been adopted to approach the NER task. Instead, *regular expressions*, and perfect matching with an existing backend database of entities (i.e., *gazetteers*) were used.

Conversely, the main novelty of this work regards the RE task. Indeed, SEED infers and disambiguates relations between previously discovered entities by exploiting the so-called "*wisdom of the crowd*", namely by using the potential of the Social Web. SEED extracts actual social events from a set of candidates that were *ranked* according to their *relevance* on the Social Web.

In a nutshell, we issued to a Web search engine a query built from the terms (i.e., mentions) representing the entities in each candidate event, and we assign to such candidate a *relevance score* that is proportional to the ranking positions of those documents returned as relevant. The rationale for this intuition is that very likely *true* events turn out to retrieve several high-ranked Web search results, thereby they may obtain high relevance scores.

Experimental results have shown that the NER module behave consistently with state-of-the-art approaches while

RE module outperforms existing solutions that usually work on a sentence-by-sentence perspective.

Possible future works we are interested in exploring concern designing a more effective NER solution and exploiting other social media – for instance nearly real-time microblogs like Twitter – to improve the performance of the RE task.

7. ACKNOWLEDGMENTS

This research was partially supported by the National PON Project TETRis - no. PON01_00451. In addition, authors would like to thank 2night S.p.A. for having provided the dataset used during the experimental phase of this work.

8. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *DL '00*, pages 85–94. ACM, 2000.

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

[3] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. *Open Information Extraction for the Web*. PhD thesis, University of Washington, 2009.

[4] S. Brin. Extracting patterns and relations from the world wide web. *The World Wide Web and Databases*, pages 172–183, 1999.

[5] R. Bunescu and R. Mooney. Subsequence kernels for relation extraction. *Advances in Neural Information Processing Systems 18*, pages 171–178, MIT Press, 2006.

[6] R. C. Bunescu and R. J. Mooney. A shortest path dependency kernel for relation extraction. In *HLT/EMNLP '05*, pages 724–731. Association for Computational Linguistics, 2005.

[7] M. E. Califf and R. J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.

[8] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. Gate: an architecture for development of robust hlt applications. In *ACL '02*, pages 168–175. Association for Computational Linguistics, 2002.

[9] D. Downey, O. Etzioni, and S. Soderland. Analysis of a probabilistic model of redundancy in unsupervised information extraction. *Artificial Intelligence*, 174(11):726–748. Elsevier Science Publishers Ltd., 2010.

[10] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, and B. Taskar. 5 probabilistic relational models. *Introduction to Statistical Relational Learning*, page 129, 2007.

[11] X. Han, L. Sun, and J. Zhao. Collective entity linking in web text: a graph-based method. In *SIGIR '11*, pages 765–774. ACM, 2011.

[12] C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(9):521–538. Elsevier Science Publishers Ltd., 1998.

[13] N. Jahan, S. Morwal, and D. Chopra. Named entity recognition in indian languages using gazetteer method and hidden markov model: A hybrid approach. *IJCSET*, March 2012.

[14] N. Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *ACL '04*, page 22. Association for Computational Linguistics, 2004.

[15] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML*, June 2001.

[16] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[17] R. Malouf et al. A comparison of algorithms for maximum entropy parameter estimation. In *COLING '02*, pages 1–7. Association for Computational Linguistics, 2002.

[18] D. Maynard, V. Tablan, C. Ursu, H. Cunningham, and Y. Wilks. Named entity recognition from diverse text types. In *RANLP '01*, pages 257–274. Association for Computational Linguistics, 2001.

[19] R. McDonald. Extracting relations from unstructured text. *Rapport technique, Department of Computer and Information Science-University of Pennsylvania*, 2005.

[20] R. Mihalcea and A. Csomai. Wikify!: linking documents to encyclopedic knowledge. In *CIKM '07*, pages 233–242. ACM, 2007.

[21] A. Mikheev, M. Moens, and C. Grover. Named entity recognition without gazetteers. In *EACL '99*, pages 1–8. Association for Computational Linguistics, 1999.

[22] S. Sarawagi. Information Extraction. *Foundations and Trends in Databases*, 1(3):261–377, March 2008.

[23] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1):233–272. Kluwer Academic Publishers, 1999.

[24] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *ML '05*, pages 896–903. ACM, 2005.

[25] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *ACL '95*, pages 189–196. Association for Computational Linguistics, 1995.

[26] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.