Online News Event Extraction for Global Crisis Surveillance

Jakub Piskorski[1], Hristo Tanev[2], Martin Atkinson[2], Eric van der Goot[2], and Vanni Zavarella[2]

[1] Polish Academy of Sciences, ul. J.K. Ordona 21, 01-237 Warszawa, Poland
Jakub.Piskorski@ipipan.waw.pl
[2] Joint Research of the European Commission, 21027 Ispra (VA), Italy
{Hristo.Tanev,Vanni.Zavarella}@ext.jrc.ec.europa.eu,
{Martin.Atkinson,Erik.VanderGoot}@jrc.ec.europa.eu

Abstract. This article presents a real-time and multilingual news event extraction system developed at the Joint Research Centre of the European Commission. It is capable of accurately and efficiently extracting violent and natural disaster events from online news. In particular, a linguistically relatively lightweight approach is deployed, in which clustered news are heavily exploited at all stages of processing. Furthermore, the technique applied for event extraction assumes the inverted-pyramid style of writing news articles, i.e., the most important parts of the story are placed in the beginning and the least important facts are left toward the end. The article focuses on the system's architecture, real-time news clustering, geo-locating and geocoding clusters, event extraction grammar development, adapting the system to the processing of new languages, cluster-level information fusion, visual event tracking, event extraction accuracy evaluation, and detecting event reporting boundaries in news article streams. This article is an extended version of [20].

Keywords: event extraction, global crisis monitoring, shallow text processing, information aggregation.

1 Introduction

A massive amount of information is now stored and exchanged via the World Wide Web, mostly in the form of free text. Therefore, natural language processing technologies which can map free text into structured data formats are becoming paramount. In particular, the proliferation of electronic news media has led to an emergence of publicly accessible news aggregation systems (e.g., *Google News*, *Yahoo! News*, *SiloBreaker*, *NewsTin* and *DayLife*) on the web for facilitating navigation through news broadcast daily worldwide. Such news aggregation systems group topically related articles into clusters and classify them according to various predefined criteria. Although such systems provide a more structured view of what is happening in the world, the amount of data requiring processing by a human remains enormous. Recently, several endeavours towards

N.T. Nguyen (Ed.): Transactions on CCI V, LNCS 6910, pp. 182–212, 2011.

developing real-time news event extraction systems have been attempted, to detect key information about events from various electronic news media and summarize this information in the form of database-like structures. In this way, such systems have been successful in providing even more compact event descriptions that combine information from different sources.

This article gives an overview of the multilingual event-extraction system developed at the Joint Research Centre of the European Commission for extracting violent and natural disaster event information from on-line news articles collected through the Internet with the Europe Media Monitor (EMM) [6,5], a web based news aggregation system, which regularly checks for updates of news articles across multiple sites in different languages. Gathering information about crisis-related events over time is an important task for better understanding conflicts and for developing global monitoring systems for automatic detection of precursors for threats in the fields of conflict and health. In particular, web news reflect trends and behaviours, which constitute a powerful data source for future event prediction. The increase in security concerns since 9/11 especially those related to terrorism have significantly boosted research on developing systems for automated event extraction from news.

Formally, the task of event extraction is to automatically identify events in free text and to derive detailed information about them, ideally identifying *Who did what to whom, when, with what methods (instruments), where and why.* Information about an event is usually represented in a so called *template* which can be seen as a attribute-value matrix. An example illustrating an event template is presented in Figure 1. Automatically extracting events is a higher-level information extraction (IE) task [3] which is not trivial due to the complexity of natural language and due to the fact that, in news, a full event description is usually scattered over several sentences and articles. In particular, event extraction relies on identifying named entities and relations between them. The research on automatic event extraction was pushed forward by the DARPA-initiated Message Understanding Conferences[1] and by the ACE (Automatic Content Extraction)[2] programme. Although, a considerable amount of work on automatic extraction of events has been reported, it still appears to be a lesser studied area in comparison to the somewhat easier tasks of named-entity and relation extraction. Precision/recall figures oscillating around 60% are considered to be a good result. Three comprehensive examples of the current functionality and capabilities of event extraction technology dealing with identification of disease outbreaks and conflict incidents are given in [12], [34] and [16]. The most recent trends and developments in this area are reported in [4]. The work most similar to ours on deploying information extraction technology for merging news events from multiple news sources covering different time periods have been reported in [17] and [31].

[1] MUC - `http://www.itl.nist.gov/iaui/894.02/related/projects/muc`
[2] ACE - `http://projects.ldc.upenn.edu/ace`

TYPE:	killing
METHOD:	shooting
ACTOR:	Americans
VICTIM:	Five Iraqis
LOCATION:	Bagdad
TIME:	12.10.2003

Fig. 1. An example of an event template derived from the sentence *Five Iraqis were shot by Americans in Bagdad on the 12th of October 2003*

Due to our requirement that the event extraction system must be multilingual and easily extensible to new domains, a linguistically relatively lightweight approach has been chosen. In particular, we take advantage of clustered news data at several stages of the event extraction process and we assume that news articles are written in the inverted-pyramid style, i.e., the most important parts of the story are placed in the beginning of the article and the least important facts are left toward the end. As a consequence of this, only a tiny fraction of each news article is analyzed. Further, a cascade of finite-state extraction grammars is deployed in order to identify event-related information. These patterns are semi-automatically acquired in a bootstrapping manner, again via utilization of clustered news data. Exploiting clustered news intuitively guarantees better precision. Since information about events is scattered over different articles single pieces of information are validated and aggregated at cluster-level. One of the main prerequisites for the ability to digest massive amounts of data in real time is efficient processing. Therefore, an in-house extraction pattern-matching engine has been developed in order to find a good trade-off between terse linguistic descriptions and efficient processing. The system has been fully operational since 2007 and supports event extraction for several languages. The results are viewed in real time via a publicly accessible web page or via *Google Earth* application. An empirical evaluation revealed acceptable accuracy and a strong application potential. Although our domain centers around security-related events, the techniques deployed in our system can be applied to other domains, e.g., tracking business-related events for risk assessment.

The rest of this article is organized as follows. First, in section 2 the architecture of our live event extraction processing chain is described. Next, section 3 gives an overview of real-time news clustering. The process of geo-locating clusters is sketched in section 4. Subsequently, section 5 describes the structure of event extraction grammars, their creation, and multilingual aspects. Further, section 6 elaborates on information fusion. Event visualization and accessing fully-fledged event descriptions generated by the system is presented in section 7. Evaluation figures are given in section 8. Section 9 presents the results of some experiments on detecting news event reporting boundaries in a stream of news articles. Finally, we end with a summary and future work directions in section 10.

2 Event Extraction Process

This section briefly describes the real-time event extraction processing chain, which is depicted in Figure 2. First, before the proper event extraction process can proceed, news articles are gathered by dedicated software for electronic media monitoring, namely the Europe Media Monitor EMM system [6] that receives 100000 news articles from 2500 news sources in 42 languages each day. Secondly, all articles are classified according to around 700 categories and then scanned in order to identify known entities (e.g., geographical references, names of known people and organizations, etc.). This information is then created as meta data for each article. Next, the articles are grouped into news clusters according to content similarity. Subsequently, each cluster is geo-located. Further, clusters describing security-related events are selected via the application of key-word based heuristics.

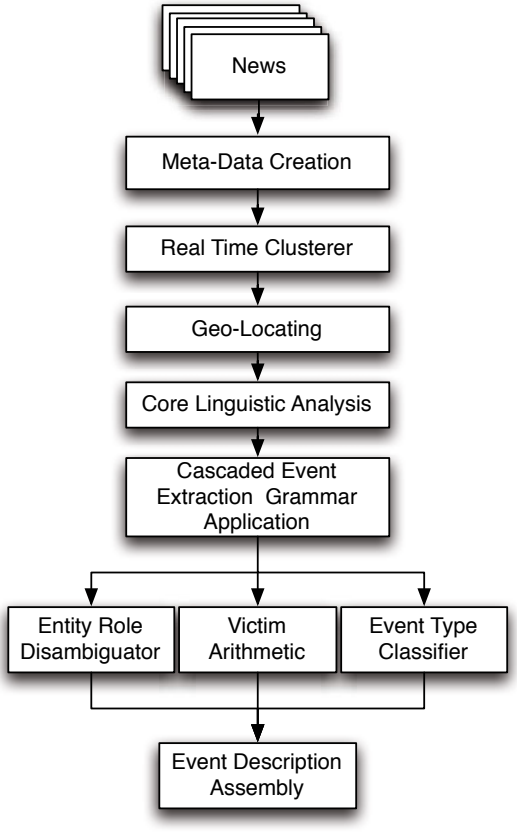

Fig. 2. Real-time event extraction processing chain

Next, each of these clusters is processed by NEXUS (News cluster Event eXtraction Using language Structures), the core event extraction engine. For each cluster it tries to detect and extract only the main event by analyzing all articles in the cluster. For each detected violent and natural disaster event NEXUS produces a frame, whose main slots are: date and location, number of killed and injured, kidnapped people, actors, and type of event. In an initial step, each article in the cluster is linguistically preprocessed in order to produce a more abstract representation of its text. This encompasses the following steps: fine-grained tokenization, sentence splitting, domain-specific dictionary look-up (e.g., recognizing numbers, quantifiers, person titles), matching of key terms indicating unnamed person groups (e.g. *civilians*, *policemen*, *Shiite*), and morphological analysis. The aforementioned tasks are accomplished by CORLEONE (Core Linguistic Entity Online Extraction), our in-house core linguistic engine [19].

Once the linguistic preprocessing is complete, a cascade of extraction grammars is applied on each article within a cluster. The patterns for the extraction grammars are created through a blend of machine learning and knowledge-based techniques. Contrary to other approaches, the learning phase is done by exploiting clustered news, which intuitively guarantees better precision of the learned patterns. The extraction patterns are matched against the first sentence and the title of each article from the cluster. By processing only the top sentence and the title, the system is more likely to capture facts about the most important event in the cluster. Figure 3 gives an example of titles and corresponding first sentences from articles reporting on a same event related to a killing of 44 people at a wedding in Turkey.

Finally, since the information about events is scattered over different articles (see the example in Figure 3), the last step consists of cross-article cluster-level information fusion, in order to produce fully-fledged event descriptions, i.e., we aggregate and validate information extracted locally from each single article in the same cluster. This process encompasses mainly three tasks, entity role disambiguation (as a result of extraction pattern application the same entity might be assigned different roles), victim counting and event type classification.

The core event-extraction process is synchronized with the real-time news article clustering system in EMM in order to keep up-to-date with most recent events.

However, in some domains, e.g., illegal migration, clustering proved to be ineffective since event reporting data seems to be more sparse, frequently coming from local news sources only, thus unlikely to form clusters. In such cases, event extraction is performed on single articles, and a larger portion of each article is analysed in order not to lose any piece of possibly relevant information [22]. Nevertheless, in this article, we focus on the description of the cluster-centric approach to event extraction, which turned out to perform good in the domain of natural disasters, violent events and other crisis-related events.

A more thorough description of the real-time news clustering, geo-locating clusters, event extraction grammars, information fusion, event reporting boundary

44 shot dead during Turkey ceremony
Eight gunmen suspected of fatally shooting 44 people at an engagement ceremony in south-east Turkey have been arrested ...
44 killed in attack in Turkey (AP)
AP - Turkish security forces on Tuesday detained eight gunmen suspected of fatally shooting 44 people, many of whom were praying, at an engagement ceremony in the rural southeast of the country ...
Turkey - Forty-four people killed at wedding party in Kurdish region
Forty-four people, including 22 women and children, were killed in an attack on a wedding party late on Monday in Turkey's Kurdish region ...
Gunmen kill at least 44 at Turkish wedding party
Masked gunmen armed with assault rifles and grenades attacked a wedding party in mainly Kurdish southeast Turkey, killing at least 44 people, authorities said ...
Masked gunmen kill 45 people at engagement party
ANKARA, Turkey – Masked assailants with grenades and automatic weapons attacked an engagement ceremony in southeast Turkey on Monday, killing 45 people ...

Fig. 3. Titles and corresponding first sentences from articles (in the same cluster) reporting on a same event related to a killing of 44 people at a wedding in Turkey

detection, issues concerning multilinguality and accessing the results produced by the live event extraction engine follows in the subsequent sections.

3 Clustering of News Articles

The real time news clustering system performs periodical hierarchical clustering on a set of news articles harvested in a 4-hour time window. A cache is maintained for twice that period in case the number of articles drops below a certain minimum, in which case the system attempts to process at least that set minimum. This process is performed every 10 minutes.

Initially each news article is considered as a cluster. The clustering process is agglomerative and uses average group linkage to determine the distances between the clusters. For calculating these distances a simple cosine measure is used. The clustering process continues until the maximum cosine distance falls below a certain set threshold (a function of the size of the term-set which can be taken as a proxy for the vector density, given a maximum word count per article).

The article feature vectors are simple word count vectors. The words used for the construction of the word-document space are selected based on various criteria depending on the amount of information available in the time window under consideration. No lemmatizing is performed and a simple bag-of-words approach is used instead.

The system maintains a constantly updated frequency table for all words found in all articles processed for a particular language over time. The system employs a technique similar to an infinite input response filter to calculate and maintain these frequencies, which can shift over time. After a fixed run-length the number of samples used to calculate the average is set. This prevents loss of accuracy, numeric overflow and gives a more realistic temporal view of the word frequencies. Additionally, a full information entropy calculation is performed over the article set in the time window every clustering run [27].

The selection criteria for the words to be used for the construction of the word-document space are as follows:

- only use words of more than 2 characters (most words of 2 or less characters do not carry any significant meaning),
- reject words that are in the 100 most frequently used words in the language (empirically this contains most of the non-significant words greater than 2 characters e.g. *the*, *and*, *for*, *that*, *said*, etc.),
- reject words that have an information entropy higher than a certain language-dependent threshold (typically 0.75) in the document set under consideration (functionally almost the same as the top 100 words removal, but now for the document set at hand removes words like days of the week, etc.),
- consider only the first 200 words (length normalisation) of each news article (most news-relevant information will be present in the first 200 words),
- only select words that appear at least twice in at least 2 documents, unless this results in a number of unique words less than a certain threshold (typically 1000) in which case only use words that appear at least once in at least 2 documents.

A cluster is only valid if it contains articles from at least 2 different sources and these articles are not duplicates.

After the clustering procedure the new clusters are compared with those calculated at $t - \Delta_t$. If there is any overlap between clusters, the articles appearing in cluster at $t - \Delta_t$ which do not appear in the current cluster, are linked to the current cluster. There are 2 possible reasons for the missing articles: either the articles have simply dropped out of the time window, or the cluster has broken up due to the changes in word-document space. This merging process can lead to internal duplication of articles, i.e. the articles are linked back into the original cluster, but also exist in a newly formed cluster. In order to avoid this duplication, the current cluster set is scanned for internal overlap and any overlapping clusters are merged.

In this way news stories consisting of large numbers of articles, far exceeding the number present in any time-window, can be tracked incrementally over time, as long as there is some reporting in the time-window.

The graph in Figure 4 shows the evolution of a cluster over time as a story. On the bottom axis is time, both left and right axis show the number of articles. The left axis shows the cumulative and 4hour sum count, while the right axis shows the instantaneous (10 minute) count. Similarly, the area in the chart shows the

cumulate count of articles in the cluster and the line in the chart shows the story evolution (number of articles present in a 4 hour window) and the bars show the instant (10 minute) entry of new articles into the cluster. Red dots on the line indicate potential breaking news moments.

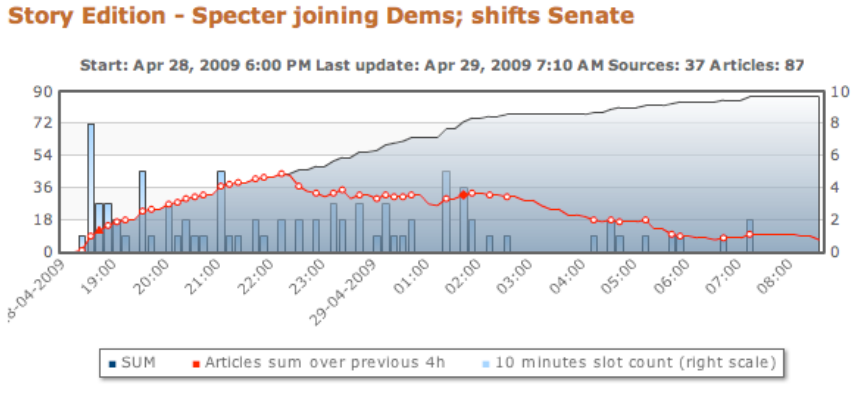

Fig. 4. Evolution of a cluster over time

4 Geo-locating

Homographs pose a well-known problem in the process of geo-tagging news articles [24]. In particular, words referring to place names may:

(a) occur as person names, e.g., *Conrad Hilton* are the names of towns in the USA, New Zealand and Canada,
(b) occur as common words, e.g., *And* is a village in Iran,
(c) have variants, e.g., well known cities have language variants, e.g., *Seul* meaning alone/only in French is also the Capital of South Korea in Portuguese and Italian,
(d) refer to different locations, e.g., there are 33 places named *Clinton*.

Additional complications are caused by language inflection and capital cities referring to the reporting location.

Our latest approach is to conduct homograph disambiguation at two distinct levels, firstly at the level of individual article and secondly, when the articles have been clustered. The article geo-tagging algorithm occurs as follows: First, the problem in (a) is solved by removing previously recognized person and organization names from the toponym candidate list. Next, a multi-lingual gazetteer of place, province, region and country names (circa 600000 entries) is used to geo-match a list of candidate locations in the news articles.

In order to disambiguate homographs that are common words and place names (see (b) and (d)), the traditional approach is to use language dependent stop

word lists. We use a different approach based on two characteristics maintained in our gazetteer. The first characteristic classifies locations based on their perceived size, such that capital cities and major cities have a higher class than small villages. The second characteristic maintains the hierarchical relation of place in its administrative located hierarchy (i.e., town, in province, in region, in country). The disambiguation algorithm lets high class locations pass through as well as locations that have a containment relation with other candidate locations (e.g., *Paris*, *Texas*, *USA*). To resolve (c) we remove any geo-tags that are incompatible with the language[3] of the article.

In order to handle name inflection, names are maintained with their variants encoded as a regular expression. Only matches, for which the language of the name and the article are compatible, are maintained in the candidate location list. Next, a Newswire location filtering is applied, where the word position of the candidate location is used to promote locations occurring after an initial location since the Newswire location generally appears earlier in the article.

We are currently experimenting with adding further filtering mechanisms coming from the semantics of the locative prepositional phrases, in which toponyms are usually nested. This seems particularly needed in domains, such as border security, that involve motion events (e.g. a person entity moving from place A and heading to place B possibly passing through place C), with typically multiple place name mentions, most of which are not the actual location of the event. In most of these contexts, a simple semantic analysis is sufficient to discard toponyms that are fulfilling irrelevant semantic roles (like the ORIGIN of the person involved in an illegal border crossing event), so as to largely reduce the complexity of the whole geo-location task.

At the level of the cluster all named entities and geo-tags in the articles of the cluster are gathered together. This is necessary to remove named entities that have not been matched in articles (a common reporting trait is to mention only well known names, e.g., *Clinton*, *Hilary Rodham Clinton*, *Hilary Rodham* and *Hilary Clinton*). Next, we gather the highest score for each place by maximum value (we don't sum same place values since often news wires are repeated many times and this would affect the real location that is cited in the article). Next, we cap the scores, here we look for tags that are either countries or regions and check whether any of the places cited are physically inside. When this is the case we increase the place score, i.e., some articles of the same cluster will talk more generally of an event in China while others will be more specific and will describe a location like Sechuan Province. Finally, the highest scoring place tag is chosen as the geo-location of the cluster.

Compared to the previous version of the geo-location algorithm [29] we have carried out a more thorough evaluation for English, French and Italian. The data set used were the daily clusters over three selected days. The results are presented in Table 1.

[3] The language of the article is primarily determined by a source related language attribute but if necessary corrected by an automatic language detector which forms part of the system.

Table 1. Precision and Recall of the Geo-Location algorithm

Lang	Cluster Size	Correct Tags	Missed Tags	False Pos.	Precision	Recall
EN	294	270	17	7	97%	94%
FR	77	76	1	4	95%	98%
IT	83	74	9	1	99%	89%

A particular system of Geo Code Identifiers (GID's) has been employed in this system. The objective of encoding the GID's is to allow fast evaluation of part of the disambiguation algorithm notably with respect to the containment functions that can be computationally intensive to evaluate using classical topographic algorithms. Here, we capitalize on the fact that our gazetteer is predefined so we use off-line processing to determine such topographic containment relations between elements in the gazetteer and higher administrative levels.

Therefore, the basic objective of encoding of the GID's is to be able to quickly determine whether two places are in the same provincial, regional and national administrative areas without having to do costly topographic calculations, index look ups or string comparisons. To fulfill this situation the GID is represented as a long integer which encodes each individual place name's administrative characteristics. In particular, country ID, regional ID, provincial ID and place name ID are represented with 10, 6, 8 and 30 bits respectively.

We start by defining GID's for our administrative area dataset, GAUL (Global Administrative Unit Layers) from the Food and Agriculture Organization of the United Nations. To encode the GAUL data set we start with all the country polygons that we assign a GID as an integer value in ascending order. For each country we then assign a sequential GID for each region inside the country again following our system outline above, then we repeat this operation for each province in each of the regions.

We then take our home grown multilingual gazetteer of place names, which has grown from a number of open source information repositories like Wikipedia and Geonames. For each location in the place name gazetteer we look for the provincial polygon that contains the place name coordinates. The GID of the provincial polygon will, by our encoding scheme, already contain the regional and national GID's. Hence we just assign a sequential number to the place name corresponding to its assignment (sequence) in the provincial polygon. Determining whether two GID's refer to places in a same country or administrative region boils down to simple bit operations.

5 Cascaded Extraction Grammars

In order to detect event information from news articles we deploy a cascade of finite-state extraction grammars. Following this approach was mainly motivated by the fact that:

– finite-state grammars can be efficiently processed, and
– using cascades of smaller grammars facilitates the maintenance of underlying linguistic resources and extending the system to new domains.

This section first introduces our in-house IE-oriented pattern matching engine. Subsequently, the structure of the current event extraction grammar for English is presented. Finally, the automatic extraction pattern acquisition and issues concerning adapting event extraction process to new languages are briefly addressed.

5.1 Pattern Matching Engine

In order to guarantee that massive amounts of textual data can be digested in real time, we have developed ExPRESS (Extraction Pattern Engine and Specification Suite), a highly efficient extraction pattern engine [18], which is capable of matching thousands of patterns against MB-sized texts within seconds. The specification language for creating extraction patterns in ExPRESS is a blend of two previously introduced IE-oriented grammar formalisms, namely JAPE (Java Annotation Pattern Engine) used in the widely-known GATE platform [9] and XTDL, a significantly more declarative and linguistically elegant formalism used in a lesser known SProUT platform [10].

An ExPRESS grammar consists of pattern-action rules. The left-hand side (LHS) of a rule (the recognition part) is a regular expression over flat feature structures (FFS), i.e., non-recursive typed feature structures (TFS) without structure sharing, where features are string-valued and unlike in XTDL types are not ordered in a hierarchy. The right-hand side (RHS) of a rule (action part) constitutes a list of FFS, which will be returned in case LHS pattern is matched.

On the LHS of a rule variables can be tailored to the string-valued attributes in order to facilitate information transport into the RHS, etc. Further, like in XTDL, functional operators (FO) are allowed on the RHSs for forming slot values and for establishing contact with the 'outer world'. The predefined set of FOs can be extended through implementing an appropriate programming interface. FOs can also be deployed as boolean-valued predicates. The two aforementioned features make ExPRESS more amenable than JAPE since writing 'native code' on the RHS of rules (common practice in JAPE) has been eliminated. Finally, we adapted JAPE's feature of associating patterns with multiple actions, i.e., producing multiple annotations (possibly nested) for a given text fragment. It is important to note that grammars can be cascaded. The following pattern for matching events, where one person is killed by another, illustrates the syntax.

```
killing-event :>
 ((person & [FULL-NAME: #name1]):killed
   key-phrase & [METHOD: #method, FORM: "passive"]
  (person & [FULL-NAME: #name2]):killer):event
 ->
 killed: victim & [NAME: #name1],
```

```
killer: actor & [NAME: #name2],
event: violence & [TYPE: "killing",
                   METHOD: #method,
                   ACTOR: #name2,
                   VICTIM: #name1,
                   ACTOR_IN_EVENTS: numEvents(#name2)]
```

The pattern matches a sequence consisting of: a structure of type `person` representing a person or group of persons who is (are) the victim, followed by a phrase (a structure of type `key-phrase`) in passive form, which triggers a *killing event*, and another structure of type `person` representing the actor. The symbol `&` links a name of the FFS's type with a list of constraints (in form of attribute-value pairs) which have to be fulfilled. The variables `#name1` and `#name2` establish bindings to the names of both humans involved in the event. Analogously, the variable `#method` establishes binding to the method of killing delivered by the `key-phrase` structure. Further, there are three labels on the LHS (`killed`, `killer`, and `event`) which specify the start/end position of the annotation actions specified on the RHS. The first two actions (triggered by the labels `killed` and `killer`) on the RHS produce FFS of type `victim` and `actor` resp., where the value of the `NAME` slot is created via accessing the variables `#name1` and `#name2`. Finally, the third action produces an FFS of type `violence`. The value of the `ACTOR_IN_EVENTS` attribute is computed via a call to a FO `numEvents()` which contacts external knowledge base to retrieve the number of events the current actor was involved in the past.

We have carried out some experiments to compare the run-time behavior of EXPRESS with XTDL. In order to do this we have used the event extraction grammar presented in Section 5.2. It was converted in almost one-to-one manner into a XTDL grammar.[4] In an experiment, the grammars were applied to a 167 MB excerpt of the news on terrorism, consisting of 122 files on a PC Pentium 4 machine with 2,79 GHz. The average run-time for processing a single document (average size of 1,37 MB) for EXPRESS and XTDL interpreter was 5.57 sec. and 58.30 sec. respectively, i.e, EXPRESS was circa 10 times faster. It is important to note that the average number of matches per document amounted to ca. 60 000 and that time given above included deployment of core linguistic processing components by both engines.[5] The time consumed by pattern matcher of EXPRESS and XTDL disregarding the time needed to deploy core linguistic processing components amounted to 50.34 sec. and 2.56 sec. respectively.

A comprehensive overview of the techniques for compiling and processing grammars and the entire ExPRESS engine as well as more detailed description

[4] It turned to be a relatively simple task since the core linguistic components provided with EXPRESS have nearly identical functionality and I/O specification as those used in SPROUT. However, some rules had to be expressed as two rules in XTDL since XTDL rules do not allow for specifying more than one output structure directly.

[5] Both engines used the same type of components, namely tokenizers, gazetteers and morphological analysis component.

of the run-time performance comparison of ExPRESS, XTDL and JAPE is given in [18].

5.2 Event Extraction Grammars

There are two different approaches to building event extraction grammars:

- the complexity of the language is represented at the level of lexical descriptions, where each word in the lexicon is provided with a rich set of semantic and syntactic features [2,23],
- the complexity of the language is represented through different, mostly linear, patterns in the grammar, which rely on superficial or less sophisticated linguistic features [32].

Providing rich lexical descriptions like verb sub-categorization frames in [2] or ontologies in [23] requires a linguistic expertize, on the other hand, more shallow lexical descriptions will result in more patterns to encode the necessary linguistic knowledge. However, superficial patterns are closer to the text data. We believe that their creation is more intuitive and easier for non-experts than building ontologies or sub-categorization frames. Writing such patterns is easier for languages like English, where the word ordering obeys strict rules and the morphological variations are not an important issue. Therefore, we followed this approach when creating our English event extraction grammar.

This grammar in its current version consists of two subgrammars. The first-level subgrammar contains patterns for recognition of proper names, e.g., person names (*Osama bin Laden*), and small-scale structures, e.g., unnamed person groups (*at least five civilians*), named person groups (*More than thousands of Iraqis*), etc. As an example consider the following rule for detecting mentions of person groups.

```
person-group :> ((gazetteer & [GTYPE: "numeral",
                               SURFACE: #quant,
                               AMOUNT: #num])
                 (gazetteer & [GTYPE: "person-modifier"])?
                 (gazetteer & [GTYPE: "person-group-proper-noun",
                               SURFACE: #name1])
                 (gazetteer & [GTYPE: "person-group-proper-noun",
                               SURFACE: #name2])):name
  -> name: person-group & [QUANTIFIER: #quant,
                           AMOUNT: #num,
                           TYPE: "UNNAMED",
                           NAME: #name,
                           RULE: "person-group"],
                         & #name = Concatenate(#name1,#name2).
```

This rule matches noun phrases (NP), which refer to people by mentioning their nationalities, religion or political group to which they belong, e.g. *three young*

Chinese, *one Iraqi Muslim*, *three young Maoists*. The words and phrases which fall in the category `person-group-proper-noun`, `numeral` and `person-modifier` (e.g. *young*) are listed in a domain-specific lexicon (gazetteer) together with a number of other semantic classes relevant for the domain. In this way the grammar rules are being kept less language dependent, i.e., rules can be applied to languages other than English, provided that language-specific dictionaries back up the grammar. As it will be shown in Section 5.4, semantic categories can be populated semi-automatically using machine learning techniques.

Through abstracting from surface forms in the rules themselves the size of the grammars can be kept relatively low and any modifications boils down to extending the lexica, which makes the development for non-experts straightforward. Further, the first-level grammar does not rely on morphological information and uses circa 40 fine-grained token types (e.g., `word-with-hyphen`, `opening-bracket`, `word-with-apostrophe`, `all-capital-letters`), which are to a large extent language independent. Consequently, the majority of the first-level grammar rules for English can be used for processing texts in other languages. For instance, the following rule for recognition of person names (a title followed by a first name and a capitalized word) is applicable to many languages.

```
person :> ((gazetteer & [GTYPE: "title",
                         SURFACE: #title)
           (gazetteer & [GTYPE: "firstName",
                         SURFACE: #fname])
           (token &     [TYPE: "alphaFirstCapitalized",
                         SURFACE: #surname])):name
  -> name: person & [NAME: #name],
                  & #name = Concatenate(#title,#fname,#surname).
```

Clearly, some of the rules might not be applicable for some languages due to the differences in syntactic structure, but they are intuitively easily modifiable.

The second-level subgrammar consists of patterns for extracting partial information on events: actors, victims, type of event, etc. Since the event extraction system is intended to process news articles which refer to security and crisis events, the second-level grammar models only domain-specific language constructions. Moreover, the event extraction system processes news clusters which contain articles about the same topic from many sources, which refer to the same event description with different linguistic expressions. This redundancy mitigates the effect of phenomena like anaphora, ellipsis and complex syntactic constructions. As mentioned earlier, the system is processing only the first sentence and the title of each article, where the main facts are summarized in a straightforward manner, usually without using coreference, sub-ordinated sentences and structurally complex phrases. Therefore, the second-level grammar models solely simple syntactic constructions via utilization of 1/2-slot extraction patterns like the ones given below. The role assignments are given in brackets.

```
[1] PERSON-GROUP <DEAD> "were killed"
[2] {PERSON | PERSON-GROUP} <DEAD> "may have perished"
```

```
[3] PERSON-GROUP <DEAD> "dead"
[4] "police nabbed" PERSON <ARRESTED>
[5] PERSON-GROUP <DISPLACED> "fled their homes"
[6] PERSON <DEAD> "was shot dead by" PERSON <PERPETRATOR>
```

These patterns are similar in spirit to the ones used in AutoSlog [26]. They are acquired semi-automatically (see 5.3). The fact that in English the word ordering is more strict and the morphology is simpler than in other languages contributes also to the coverage and accuracy of the patterns, which encode non-sophisticated event-description phrases.

To sum up, event extraction system discards the text, which goes beyond the first sentence for the following reasons:

- handling more complex language phenomena is hard and might require knowledge-intensive processing,
- the most crucial information we seek for is included in the title or first sentence,
- if some crucial information has not been captured from one article in the cluster, we might extract it from other article in the same cluster.

In order to keep the grammar concise and as much as possible language independent, we represent in the grammar all surface-level linear patterns via pattern types, which indicate the position of the pattern with respect to the slot to be filled (left or right). To be more precise, all patterns are stored in a domain-specific lexicon (applied prior to grammar application), where surface patterns are associated with their type, the event-specific semantic role assigned to the entity which fills the slot (e.g., `WOUNDED`, `KIDNAPPED`, `DISPLACED`, etc.) and the number of the phrase which may fill the slot (singular, plural, or both). For instance, the surface pattern `"shot to death"` for recognizing dead victims as a result of shooting, is encoded as follows.

```
shot to death [TYPE: right-context-sg-and-pl,
               SURFACE: "shot to death",
               SLOTTYPE: DEAD]
```

The value of the `TYPE` attribute indicates that the pattern is on the right-hand-side of its slot (`right-context`), which can be filled by a phrase which refers to one or many people (`sg-and-pl`). The event-specific semantic role, assigned to each NP filling the slot is `DEAD`. Via such an encoding of event-triggering linear patterns, the extraction patterns [1] and [3] from the list given above are merged into one (in a simplified form):

```
dead-person :> person-group
               gazetteer & [TYPE: right-context-sg-and-pl
                            SLOT: dead]
      -> ...
```

Interestingly, we have found circa 3000 event-triggering domain-specific surface patterns for English. Clearly, the strict word ordering and relatively simple morphology allowed for easy generation of pattern variants.

5.3 Pattern Acquisition

For creating second-level grammar patterns described in previous section a weakly supervised machine learning (ML) algorithm has been deployed. It is similar in spirit to the bootstrapping algorithms described in [15,33]. In particular, the pattern acquisition process involves multiple consecutive iterations of ML followed by manual validation. Learning patterns for each event-specific semantic role requires a separate cycle of learning iterations. The method uses clusters of news articles produced by EMM. Each cluster includes articles from different sources about the same news story. Therefore, we assume that each entity appears in the same semantic role in the context of one cluster ('one sense per discourse' [11]). The core steps of the pattern acquisition algorithm are as follows:

1. Annotate a small corpus with event-specific information,
2. Learn automatically single-slot extraction patterns from annotated corpus,
3. Manually validate/modify these patterns,
4. If the size of the pattern set exceeds certain threshold, then terminate,
5. Match the patterns against the full corpus or part of it,
6. Assign semantic roles to entities which fill the slots,
7. Annotate automatically all the occurrences of these entities in all articles in the same cluster with the corresponding role this entity has been assigned in the previous step and goto 2.

An automatic procedure for syntactic expansion complements the learning, i.e., based on a manually provided list of words which have identical (or nearly identical) syntactic model of use (e.g. *killed*, *assassinated*, *murdered*, etc.) new patterns are generated from the old ones by substituting for each other the words in the list. Subsequently, some of the automatically acquired single-slot patterns were used to manually create 2-slot patterns like `X shot Y`. The pattern acquisition process is described thoroughly in [28].

5.4 Semi-supervised Learning of Semantic Categories

The other main step towards language and domain customization of the lexical resources deployed by extraction grammar is the population of the domain-specific semantic classes referenced by grammar rules. The task is also partially automated through application of machine learning techniques.

A semi-supervised system, *Ontopopulis*, is deployed which takes as input a small set of sample phrases belonging to a target semantic class (seed terms) and a unannotated corpus of news article, and then further populates the semantic class with new instances (terms) [30].

The learning process consists of two steps: a 'Feature extraction and weighting' stage, where context features for a category C are selected among the uni-grams and bi-grams co-occurring with any of the seed terms of C, and then weighted and ordered based on a Pointwise Mutual Information measure; a 'Term selection and scoring' stage exploiting a term feature vector representation, based on co-occurrence between candidate terms and features. The whole learning process

is semi-supervised, as a minimal human intervention is required in the form of manual selection of context feature output between the two stages.

An evaluation of the learning accuracy performance was carried out for Spanish and Portuguese language on a number of semantic categories, such as Person, Weapon, Crime, Infrastructure,etc. As an example, for the classes Person and Weapon in Spanish, term precision scored 71% and 14%, respectively, while precision on the 20 top ranked terms raised to 95% and 60%, with seed set size of 56 and 22 and generative factors of 7.2 and 5.6, respectively. This shows that, while absolute accuracy is highly dependent on the target category, the tool is generally able to properly rank the learned terms based on their semantics.

5.5 Multilinguality

The English event extraction grammar described in the previous section relies mainly on a multilingual named-entity recognition grammar, language-specific dictionary of NE-relevant trigger words, and a language-specific dictionary of surface-level event extraction patterns. In this way adapting the current event extraction grammar to new languages does not require much linguistic expertise.

We have extended the system to the processing of Italian, Spanish, Portuguese, French, Russian and recently Arabic through providing the aforementioned language-specific resources. Although the system's precision was as high as expected, an empirical coverage analysis for Italian showed that there are some drawbacks when surface-level patterns are applied. This is mainly due to the free-word order and relative morphological richness of Romance languages. Further, Italian appears to be more verbose with the respect to expressing information about events, i.e., it is structurally more complex. For instance, Italian verb phrases describing events show some additional structural complexity at the linear level due to the encoding of some essential information on the event, like the 'means' or 'instruments' of a killing act, in prepositional phrases, as opposed to English which typically uses lexical content of the main verb itself to convey such information. This is shown in the following sample excerpts from Italian and English articles from cross-lingual clusters about the same event returned by EMM news aggregation system.

> *excite-news Monday, May 19, 2008 10:43:00 AM CEST (ANSA) - MANILA, 19 MAG - Un uomo* **ha crivellato a colpi di mitra** *alcune case di Calamba, una citta' vicino Manila, uccidendo otto persone....*

> *cnn Monday, May 19, 2008 5:58:00 AM CEST A man* **strafed** *several houses during a shooting spree early Monday in a town south of Manila, killing eight people and wounding six others...*

Consequently, we designed a slightly more linguistically sophisticated rules for Italian, Spanish and Portuguese, which cover a rich variety of morphological and

syntactic constructions. This yielded better overall extraction accuracy. Some evaluation figures are given in Section 8, whereas the details of the process of adapting the system to the processing of Italian are presented in [35]. While we applied the same linguistic approach for Spanish and Portuguese, in [30] we performed a preliminary evaluation on a minimal, machine learning-oriented customization to these two languages, consisting of only a slight modification of the Italian person entity recognition grammar, together with the application of the linear pattern and semantic category learning methods introduced above. Although not impressive in absolute terms, the performance figures in both languages show a significant improvement with respect to the corresponding baseline systems, proving the effectiveness of the customization methods in addressing the multilinguality challenge.

Although we are currently developing linguistic resources for tackling the event extraction task for Arabic (EMM has some 86 different news sources in this language) and apply similar techniques as for English, our initial approach was slightly different. First, statistical machine translation systems were used for translating the articles in the Arabic clusters into English. Next, these translations were passed into the English event extraction system. In particular, we integrated two Arabic-to-English translation systems into the live processing chain, namely *Google Translate*[6] and *LanguageWeaver*[7]. We plan to compare the results of this approach with a first release of a full-fledged event extraction system for Arabic.

6 Information Fusion

Once the event extraction grammars has been applied locally at document level the single pieces of information are merged into fully-fledged event descriptions. In particular, three tasks are performed at this level: (a) semantic role disambiguation, (b) victim counting, and (c) event type classification. They are described briefly.

Semantic Role Disambiguation: If one and the same entity has two roles assigned in the same cluster, a preference is given to the role assigned by the most reliable group of patterns. The double-slot patterns are considered the most reliable. Regarding the one-slot constructions, patterns for detection of *killed*, *wounded*, and *kidnapped* are considered as more reliable than the ones for extraction of the *actor (perpetrator)* slot (the latter one being more generic). The pattern reliability ranking is based on empirical observations.

Victim Counting: Another ambiguity arises from the contradictory information which news sources give about the number of victims (e.g., killed) An ad-hoc algorithm for computing the most probable estimation for these numbers is applied. It finds the largest group of numbers which are close to each other and

[6] `http://translate.google.com/translate_t?hl=en`
[7] `http://www.languageweaver.com`

subsequently finds the number closest to their average. All articles, which report on number of victims which significantly differs from the estimated cluster-level victim number are discarded. In order to perform victim arithmetics at the document level a small taxonomy of person classes [29] is used. The article-level victim counting algorithm consists of the following steps:

1. Extract from a news article a set of NPs $E = \{E_1, E_2, \ldots, E_n\}$, which refer to individuals or groups with definite number of people and which have certain semantic role.
2. Map each E_i to a concept from the taxonomy using a list of keyword-taxonomy mappings (we denote it as $conc(E_i)$). As an example consider the text *One journalist was beaten to death and two soldiers died in the clashes in which four civilians lost their lives in total*. Three NPs referring to killed people would be extracted: *one journalist*, *two soldiers* and *four civilians*, and mapped to the taxonomy categories `journalist`, `serviceman`, `civilian`.
3. Delete each E_i from E, if there exist $E_j \in E$ such that $conc(E_i)$ *is*-a $conc(E_j)$, i.e., $conc(E_i)$ is a direct or indirect successor of $conc(E_j)$ in the taxonomy. In this manner only the most generic NPs are left in E. In our example we may identify one such relation, namely, *journalist is* a *civilian*, therefore we delete the NP *one journalist* from E.
4. Sum up the numbers reported from the NPs in E. In our example we have to sum up the numbers from the phrases *four civilians* and *two soldiers* (four and two).

Event Type Classification: Whenever possible, we assign a class label to the detected violent event or disaster. Some of the most used event classes are *Terrorist Attack*, *Bombing*, *Shooting*, *Air Attack*, *Man-Made Disaster*, *Floods*, etc. The event type classification algorithm uses a blend of keyword matching, taxonomy of event types and domain specific rules.

First, all potential event types are assigned ranks based on the matching of type-specific keywords in the articles in a given cluster (a rank for a given type is set to zero if no keyword related to this type was matched in the articles in the given cluster). Next, a non-zero valued rank of a more specific event type is boosted in case the rank of the type which subsumes it is non zero. The logic behind this step is that when some event takes place, it can be referred to at different levels of abstraction and the most specific reference should be preferred, since it is the most informative. Finally, the type with the highest rank is selected, unless some domain-specific event type classification rule (they have higher precedence) can be applied. As an example, consider the following domain specific rule: if the event description includes named entities, which are assigned the semantic role *Kidnapped*, as well as entities which are assigned the semantic role *Released*, then the type of the event is *Hostage Release*, rather than *Kidnapping*. Such rules were introduced based on empirical observations.

It is important to note that the information fusion algorithm has some limitations since it considers only one main event per news cluster, ignoring events of lower importance or incidents subsumed by the main event.

We recently replaced the simple keyword matching mechanism initially deployed with a more expressive regular expression pattern matching engine, powered with boolean operators. Experiments on French and Italian language indicate a significant comparative advantage of boolean combinations of patterns over simple keywords in terms of accuracy.

7 Live Event Tracking

Multilingual crisis-related event tracking poses a number of practical issues, mainly related to the correct geo-spatial visualization of the event together with it's principal characteristics. Another concern is to minimize constraints on end users to rely on expensive and proprietary desktop applications. We fulfill these issues by publishing the event data using current Internet standard formats, namely, KML and GeoRSS. In particular, the results of the event extraction are accessible in two ways:

- via *Google Earth* application which is passed event descriptions in KML format[8],
- via a publicly accessible web client[9] that exploits the *Google Maps* technologies and connects to our KML server

The event description interface provides elements in two languages. The title, description and the precise location of the event is presented in the native language of the event, i.e., the language of the cluster being processed. In the English language we provide the event consequences and the relative geographical path to the place. The diagram in Figure 5 shows the Google Earth user interface for French, Arabic and Italian respectively.

To provide immediate clues on the cause and effect of the event we use three visual indicators: the icon image of the event type that is geo-located on the map; the size of the icon depends on the size of the cluster; and the magnitude of the consequence of the event is indicated by a colored circle around the event. A key to the symbols used in our system is given in Figure 6.

The image in Figure 7 shows the Google Maps representation of the results of the event extraction system (in Russian). The balloon shows a heat wave event that occurred in Moscow on 28 April 2009.

8 Event Extraction Evaluation

An evaluation of the event extraction performance has been carried out on 368 English-language news clusters based on news articles downloaded on 24 January 2008. 29 violent events are described in these clusters and 27 out them were

[8] For English: start Google Earth application with KML: `http://press.jrc.it/geo?type=event&format=kml&language=en`. For other languages change the value of the language attribute accordingly.

[9] `http://press.jrc.it/geo?type=event&format=html&language=en`

Fig. 5. Event visualization in Google Earth: Diagram showing events detected in French, Arabic and Italian. Note that the event meta data is expressed in English whilst the original, title and description are maintained.

detected by our system (**93% coverage**). In some cases several clusters referred to the same event. We consider that an event is detected by the system, if at least one cluster referring to it was captured.

Our system detected 55 news clusters which refer to 37 violent and non-violent events. 27 out of these 37 detected events are violent events (**73% precision**). In the context of crisis monitoring, discovery of disasters causing victims may also be considered relevant. Our system detected 3 man made disasters; if we consider them relevant together with the violent events, then the precision of the event extraction becomes **81%**. With respect to victim counting, for the number of dead and injured an accuracy of 80% and 93% could be achieved respectively. The event type detection performs significantly worse, i.e., only in 57% of the cases the correct event type could be assigned.

One of the most frequent errors in event classification was in case of clusters referring to terrorist bombing, which were not classified as *Terrorist Attack*, but simply as a *Bombing*. Another problem is that classification based on simple keyword matching sometimes gives wrong results, e.g., an airplane crash was

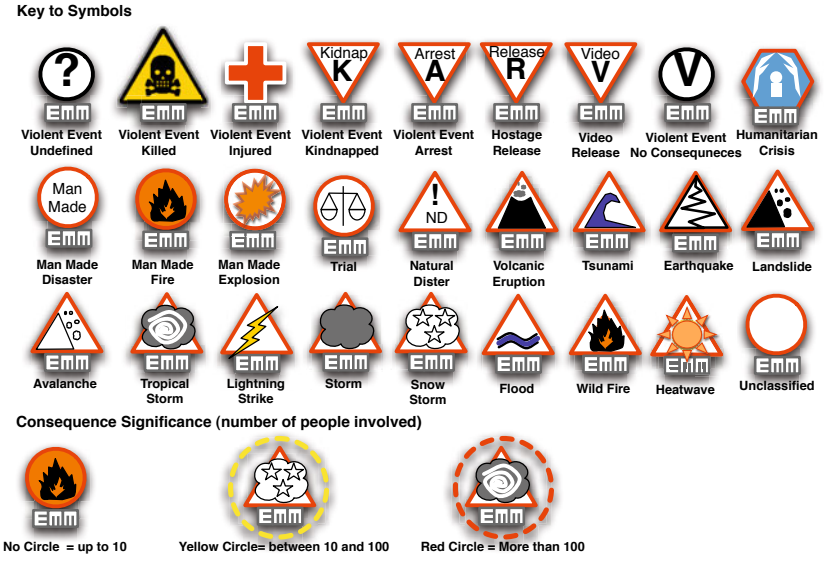

Fig. 6. Key to event type icons and magnitude indicators

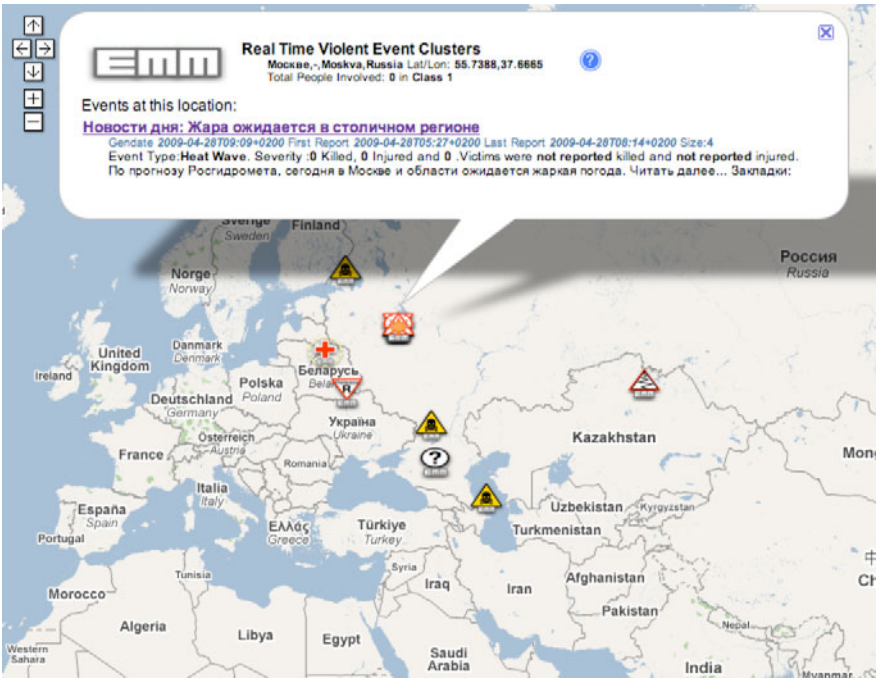

Fig. 7. Event visualization in Google Maps: The balloon shows a heat wave event that occurred in Moscow on 28 April 2009

classified as *Air Attack*, since a military plane was involved. Further, there are situations, in which two different incidents are reported as a part of one event. For example, a terrorist attack might include bombing and shooting. In such cases only one label is assigned to the whole event, although two labels would be more appropriate.

We have repeated the evaluation of the event extraction on several days, which yielded on an average similar accuracy figures to the ones reported above.

In order to evaluate the coverage and accuracy of the Italian event extraction system, we tested it on 213 clusters of news articles gathered during 4 consecutive days in July, 2008. Only part of the clusters were related to violent event stories. On this corpus, we ran both the baseline version of the system, namely the one based on linear patterns, and a hybrid version containing the more abstract rules (see Section 5.5) together with a subset of highly irregular linear patterns, which were not converted into more abstract rules. The rationale for this is that we deploy abstract rules in order to deal with productive, structurally complex language constructions, while backing off to surface level patterns to cover more idiomatic constructions. Table 2 shows a comparative evaluation of the two Italian event extraction systems. In particular, we measured precision, recall and F-measure for each role. For the sake of clarity, we define formally these measures. For a given slot type (role) T, let s_T be the number of slot values (answers) returned by the system, and let sc_T denote the number of correct answers (correct slot values) extracted by the system for the role T. Next, let all_T be the number of slot values annotated by a human expert as a value (answer) for the role T in the whole test corpus (ground truth). Then, the *recall*, *precision* for role T are defined as: $recall_T = |sc_T|/|all_T|$ and $precision_T = |sc_T|/|s_T|$. *F-measure* is defined as the mean of precision and recall, i.e. $F_T = (2 \cdot precision_T \cdot recall_T)/(precision_T + recall_T)$. The results of the evaluation are given in Table 2.

A clear gain in recall could be observed. A thorough analysis revealed that it was mainly due to deploying morphological abstraction and the productivity achieved through more abstract patterns. In particular, one case was interesting. *Actor* role fillers are usually the hardest to be extracted from Italian text, as their relationship with the main event verb is either to be derived through deep semantic or even pragmatic inferences, or extracted through complex two-slot patterns. Precision gain was smaller. In one particular case a deterioration in precision could be observed, namely in the case of *Arrested* role. Further details of the evaluation for Italian are given in [35]. Some evaluation figures for Spanish and Portuguese are presented in [30]

9 Event Reporting Boundaries

The major goal of applying an event extraction engine on top of EMM news clusters is to gather and update information about crisis-related events over time. Therefore, for each news story the events extracted by NEXUS at different time points are stored in a database. One of the major problems in this context is to detect duplicate events or to put it in other words, to detect new events in

Table 2. Comparison of the extraction accuracy for Italian for different roles with the baseline grammar (BL) and linguistically rich grammar (LR). P, R and F_1 stand for precision, recall and F-measure.

	Dead		*Injured*		*Actor*		*Arrested*	
	BL	LR	BL	LR	BL	LR	BL	LR
P	0.82	**0.91**	0.66	**0.77**	0	**0.66**	**0.83**	0.70
R	0.48	**0.84**	0.15	**0.53**	0	**0.25**	0.55	**0.66**
F_1	0.60	**0.87**	0.24	**0.62**	0	**0.36**	0.66	**0.67**

a news story. We consider a news story to be a sequence of news article clusters, where each such cluster contains news articles on a given topic gathered within a certain time window as the story evolves (see Section 3). Each such cluster is also associated with an event description generated by NEXUS for this particular cluster.

9.1 New Event Detection Techniques

We have explored several techniques for detecting new events in a given news story. First, we experimented with a relatively simple method, which computes the similarity between a currently extracted event and a previously extracted one (in a previous iteration) using an event similarity metric based on the content similarity of the corresponding slot values in the events being compared. In case the similarity is below a certain threshold the two events are considered to be distinct. Subsequently, this method was extended by introducing additional constraints on the confidence with which the current and past events were extracted by NEXUS. Confidence is simply measured in terms of supporting documents, i.e. documents for which NEXUS could return an event description compared with the total number of articles in the cluster. Further fine-tuning consisted of introducing some global constraints, e.g. the similarity between the current event and the average event similarity within a whole story has to be lower than a certain pre-defined threshold, otherwise the current event is considered to be a new event. We refer to the techniques described in this paragraph as OVERLAP.

Next, we investigated whether utilization of a curvature-based approach for topical segmentation of a stream of texts can improve the performance of new event detection. The basic idea behind such approaches to topic development analysis [7,13,1,25] is to compute for a given sequence of text entries $t_1, t_2, \ldots, t_k$ a sequence $s_{1,2}, s_{2,3}, \ldots, s_{k-1,k}$, where $s_{i-1,i}$ is the similarity between t_{i-1} and t_i. The latter sequence is called *similarity curve* and reflects how rapidly the content of the text stream changes between consecutive entries. In order to identify topically coherent segments one can use local minima of the similarity curve as indicators of segment boundaries, i.e. topic shift occurs at local minima. In the context of the new event detection task, our idea is to compute such a similarity curve for a given news story, and to use it for validating the decisions on tagging events as new, etc. In particular, for a given news story we

compute the similarity curve for the text sequence $t_1, t_2, \ldots, t_k$, where t_i consists of the titles and first sentences of the new articles in the story cluster at time i, i.e. articles, which appeared for the first time at iteration i.[10] Subsequently, events extracted by NEXUS are tagged as new only if certain patterns in the similarity curve (indicating a topic change) can be observed (e.g., local minima). Although the briefly sketched idea of classical curvature-based algorithms for topic development analysis is effective, we have not used it for two reasons:

- it performs best in the case of text streams with 'clear' changes of topics, which is not necessarily the case in the context of news stories oscillating around one or two major topics, and
- it captures solely the pairwise relationship between two adjacent entries in the text stream, whereas capturing the global relations between entries in the text stream would provide more useful and fine-grained information in topic development.[11]

Therefore, we applied a different algorithm for computing the similarity curve, namely the recently introduced CUTS algorithm presented in [25], which captures global relations between text entries in the input stream of texts. The sequence of text entries is mapped to a curve, which reflects the topic development patterns in terms of dissimilarity between neighbouring text entries. Unlike other curvature-based topical text segmentation algorithms, which utilize solely information of pairwise (dis)similarity between adjacent entries in the input sequence, CUTS algorithm exploits broader context, i.e. 'content' dissimilarities for all pairs of entries in the input. To be more precise, the initial sequence of text entries is mapped to a 1-dimensional space so that distances between points best match the dissimilarities between the corresponding entries. The aforementioned mapping is computed through the application of a multidimensional scaling (MDS) technique [8] applied on the dissimilarity matrix for the input sequence. Next, the time dimension is added, i.e. each point in time – index of the entry in the input sequence (x-axis) is mapped to the corresponding MDS-computed value in the 1 dimensional space (y-axis). The resulting curve is called CUTS curve. A 'stable' topic in CUTS curve corresponds to an almost horizontal segment (*dominated segment*). A smooth transition from one topic to another is reflected in the CUTS curve as a sloping curve (*drifting segment*), where the gradient angle measures how fast one topic fades out and new topic fades in. Finally, interruptions, i.e. sudden and temporary introduction of a new and significantly different topic, correspond to segments with a saw tooth shape. Interruptions may either occur within dominated or drifting segments. Figure 8 illustrates a drifting segment and a dominated segment with an interruption in CUTS curves computed for some of the news stories computed by EMM.

[10] In case of iterations in which there are no new articles, we consider for the computation of t_i the documents which were most recently added to the story cluster.

[11] Different news media might report on the same event at different times and even the same media might re-report on the same event from time to time.

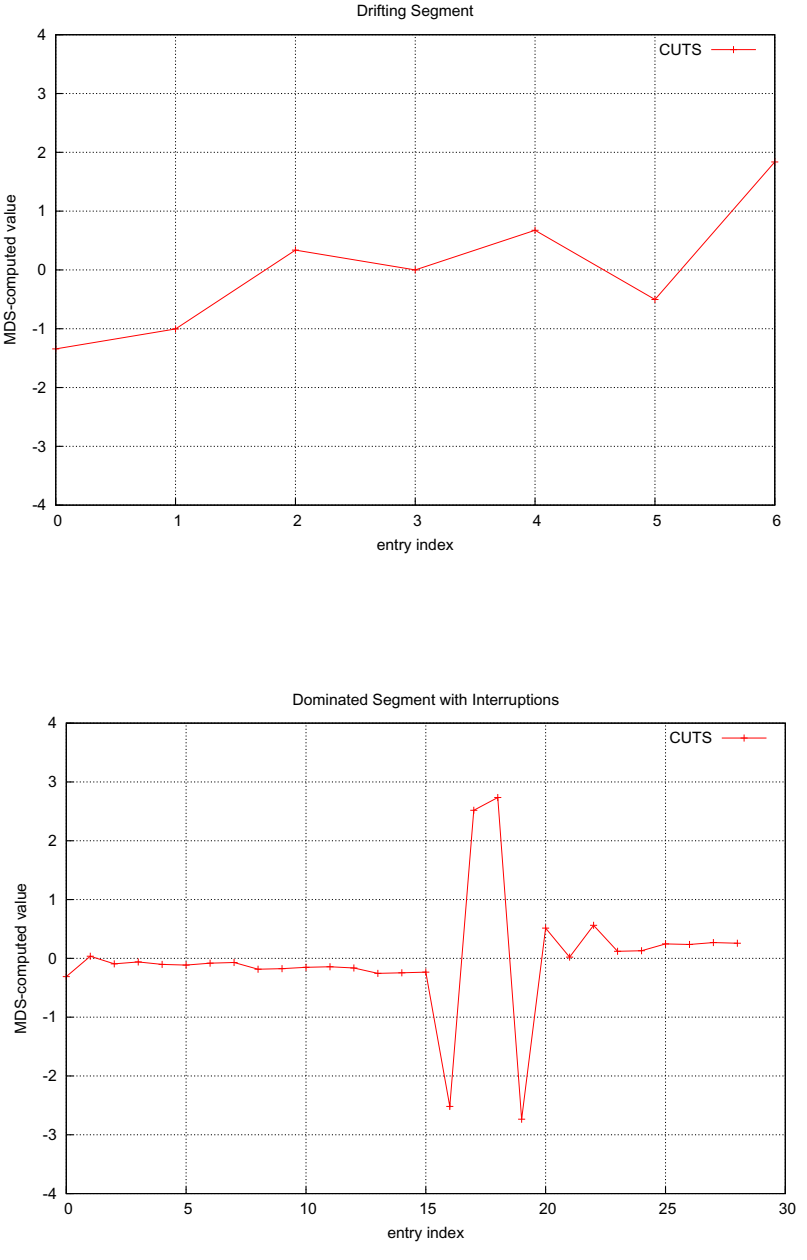

Fig. 8. An example of a drifting segment (top) and a dominated segment with an interruption (bottom) The interruption in the interval $(15, 20)$ refers most likely to a new event or possibly to some noise in the news story cluster in this interval.

The main idea behind the CUTS-based algorithm for new event detection is that predominantly interrupted or drifting segments in the CUTS curve occur directly prior to the new event. In other words, the number of articles on a new topic or aspect of the main event of the story has reached a critical mass. Since many news stories are not stable in the initial iterations, the exact conditions (for the CUTS curve), which have to be fulfilled are slightly different for the beginning of the story and subsequent iterations, e.g., the threshold for the gradient angle in the drifting segment in order to classify an event as new is higher for the beginning of the story than for subsequent iterations.

Finally, we have combined the simple algorithms described at the beginning of this section (OVERLAP) with the CUTS algorithm (we denote this combination as OVERLAP+CUTS). To be more precise, the CUTS constraints are embedded in the OVERLAP algorithms as additional ones. Furthermore, in case the original constraints of OVERLAP algorithms do not hold, but 'almost' hold (values are within a certain distance to the thresholds, etc.) the CUTS constraints are checked, and if they hold the event is tagged as a new one. A more detailed description of the algorithms for detecting event reporting boundaries are given in [21].

9.2 Evaluation

We have carried out an evaluation of the different new event detection techniques described in the previous section. For this purpose, we have collected test data as follows. NEXUS was applied at 30 minute intervals on the English news clusters produced by EMM between 6 and 16 October 2008. In this time period 6664 different news stories were collected, where 1296 of them were related to violent and natural disaster events. About one third of the latter (442) were dynamic, i.e. at least two distinct event descriptions were generated for such stories. We have applied the different algorithms on a subset thereof, consisting of 125 stories (in which the event type changed) and we have evaluated their precision, recall and F-measure. Let E be the set of automatically detected new events in the test corpus described above and let E^* be the set of events annotated as new events by a human expert. We call the latter set the ground truth. The *recall* is the ratio of correctly detected new events and all events in the ground truth, i.e. $recall = |E \cap E^*|/|E^*|$. We define *precision* as the fraction of events, which were correctly tagged by the algorithm as new events, i.e. $precision = |E \cap E^*|/|E|$. *F-measure* is defined as the mean of precision and recall, i.e. $F{-}measure = (2 \cdot precision \cdot recall)/(precision + recall)$. Table 3 presents the top results obtained for precision, recall and F-measure separately. While integration of CUTS-based constraints into simple methods resulted in a significant improvement in precision, the simple methods score best in terms of recall and F-measure. As can be observed from the table there is much room for improvement. A more thorough discussion on this topic is given in [21].

Table 3. Top results obtained for detecting new events with different techniques. P, R and F_1 stand for precision, recall and F-measure.

method	P	R	F_1
OVERLAP	0.516	**0.769**	**0.615**
CUTS	0.4	0.385	0.392
OVERLAP+CUTS	**0.75**	0654	0.583

10 Summary

This article presented a real-time and multilingual news event extraction system developed at the Joint Research Centre of the European Commission. Currently, it is capable of efficiently processing news in English, Italian, French, Spanish, Portuguese, Russian and Arabic, which provide descriptions of the latest crisis-related events around the world with a 10-minute delay. The results of the evaluation on violent and natural disaster event extraction show satisfactory performance and strong application potential for real-time global crisis monitoring. In order to guarantee ease in maintenance and extensibility to new languages and domains a shallow text processing strategy has been chosen, which utilizes clustered data at different strata. The coverage and the level of sophistication of language resources used in the system varies from language to language. The results of the live event extraction are provided via a publicly accessible web page and can also be accessed with the *Google Earth* application. The system has been fully implemented in JAVA.

In order to improve the quality of the extracted event descriptions, several system extensions are envisaged. Firstly, we are working towards fine grained classification of natural and man made disasters. We also plan to improve the classification accuracy for violent events. Secondly, we aim at extending the system to new languages. This is feasible, since our algorithms and grammars are mostly language independent. In particular, we currently focus on adapting the system to the processing of Arabic news. Being able to extract information about same event in different languages and cross-linking them adds an additional navigation layer and an opportunity for fact validation and improving the accuracy of the system. Therefore, we plan to explore techniques for refining the results of event extraction, which are similar in spirit to those presented in [14]. Furthermore, we intend to study the usability of event descriptions collected over time for the same purpose. The experiments on detecting event reporting boundaries briefly reported in this article and in [21] show that there is a lot of space for improvement. Therefore, we will pursue this line of research. Next, a long-term goal is to automatically discover structure of events and relations between them, i.e., discovering sub-events or related events of the main one, as well as implementing techniques for event duplicate detection.

The range of events the system already detects is relatively wide, which makes the tool very attractive for application in various scenarios. Nevertheless, the system is currently being customized to new domains, in particular, to detect

border-security related events (e.g., illegal border crossings and related cross-border criminal activities) [22] and outbreaks of infectious diseases in the public health domain. Although the accuracy of detecting certain types of events is quite satisfactory we can not fully eliminate human moderation in the process of gathering event information in a database. In order to facilitate this task a graphical interface has been implemented that allows users to correct, extend, merge, translate and store automatically extracted event information on events [22] in an event store – a specific knowledge repository on events. Finally, some work on improving geo-locating events through deployment of shallow linguistic analysis (along the lines sketched in section 4) is envisaged too.

Acknowledgments. The work presented in this article was supported by the Europe Media Monitoring (EMM) Project carried out by the Open Source Text Information Mining and Analysis Action in the Joint Research Centre of the European Commission. We are indebted to all our EMM colleagues without whom the presented work could not have been be possible.

References

1. Andrews, P.: Semantic topic extraction and segmentation for efficient document visualization' Master's thesis, School of Computer & Communication Sciences, Swiss Federal Institute of Technology, Lausanne (2004)
2. Aone, C., Santacruz, M.: REES: A Large-Scale Relation and Event Extraction System. In: Proceedings of ANLP 2000, 6th Applied Natural Language Processing Conference, Seattle, Washington, USA (2000)
3. Appelt, D.: Introduction to Information Extraction Technology. In: Tutorial held at IJCAI 1999, Stockholm, Sweden (1999)
4. Ashish, N., Appelt, D., Freitag, D., Zelenko, D.: Proceedings of the workshop on Event Extraction and Synthesis', held in Conjunction with the AAAI 2006 Conference, Menlo Park, California, USA (2006)
5. Atkinson, M., Van der Goot, E.: Near Real Time Information Mining in Multilingual News. In: Proceedings of the 18th World Wide Web Conference, Madrid, Spain (2009)
6. Best, C., Van der Goot, E., Blackler, K., Garcia, T., Horby, D.: Europe Media Monitor, Technical Report, EUR 22173 EN, European Commission (2005)
7. Brants, T., Chen, F., Farahat, A.: A System for New Event Detection. In: Proceedings of the 26tth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA (2003)
8. Cox, T., Cox, M.: Multidimensional Scaling. In: Monographs on Statistics and Applied Probability, 2nd edn., Chapman & Hall, London (2001)
9. Cunningham, H., Maynard, D., Tablan, V.: Jape: a Java Annotation Patterns Engine. Technical Report, CS–00–10, University of Sheffield, Department of Computer Science (2000)
10. Drożdżyński, W., Krieger, H.-U., Piskorski, J., Schäfer, U., Xu, F.: Shallow Processing with Unification and Typed Feature Structures — Foundations and Applications. Künstliche Intelligenz 1 (2004)

11. Gale, W., Church, K., Yarowsky, D.: One sense per discourse. In: HLT 1991: Proceedings of the workshop on Speech and Natural Language, Harriman, New York, USA (1992)
12. Grishman, R., Huttunen, S., Yangarber, R.: Real-time Event Extraction for Infectious Disease Outbreaks. In: Proceedings of Human Language Technology Conference 2002, San Diego, USA (2002)
13. Hearst, M.: Subtopic structuring for full-length document access. In: Postproceedings of SIGIR (1993)
14. Ji, H., Grishman, R.: Refining Event Extraction through Unsupervised Cross-document Inference. In: Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Columbus, Ohio, USA (2008)
15. Jones, R., McCallum, A., Nigam, K., Riloff, E.: Bootstrapping for Text Learning Tasks. In: Proceedings of IJCAI 1999 Workshop on Text Mining, Stockholm, Sweden (1999)
16. King, G., Lowe, W.: An Automated Information Extraction Tool For International Conflict Data with Performance as Good as Human Coders: A Rare Events Evaluation Design. International Organization 57, 617–642 (2003)
17. Naughton, M., Kushmerick, N., Carthy, J.: Event Extraction from Heterogeneous News Sources. In: Proceedings of the AAAI 2006 Workshop on Event Extraction and Synthesis, Menlo Park, California, USA (2006)
18. Piskorski, J.: ExPRESS – Extraction Pattern Recognition Engine and Specification Suite. In: Proceedings of the International Workshop Finite-State Methods and Natural language Processing 2007 (FSMNLP 2007), Potsdam, Germany (2007)
19. Piskorski, J.: CORLEONE – Core Linguistic Entity Online Extraction, Technical Report, EN 23393, Joint Research Center of the European Commission, Ispra, Italy (2008)
20. Piskorski, J., Tanev, H., Atkinson, M., Van der Goot, E.: Cluster-Centric Approach to News Event Extraction. In: Proceedings of the International Conference on Multimedia & Network Information Systems. IOS Press, Poland (2009)
21. Piskorski, J.: Exploring Curvature-based Topic Development Analysis for Detecting Event Reporting Boundaries. In: Marciniak, M., Mykowiecka, A. (eds.) Aspects of Natural Language Processing. LNCS, vol. 5070, pp. 311–331. Springer, Heidelberg (2009)
22. Piskorski, J., Atkinson, M., Belyaeva, J., Zavarella, V., Huttunen, S., Yangarber, R.: Real-Time Text Mining in Multilingual News for the Creation of a Pre-frontier Intelligence Picture. In: Proceedings of the 16th Conference on Knowledge Discovery and Data Mining (KDD 2010). ACM SIGKDD Workshop on Intelligence and Security Informatics, Washington DC, USA (2010)
23. Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., Kirilov, A., Goranov, M.: Towards Semantic Web Information Extraction. In: Proceedings of International Semantic Web Conference, Sundial Resort, Florida, USA (2003)
24. Pouliquen, B., Kimler, M., Steinberger, R., Ignat, C., Oellinger, T., Blackler, K., Fuart, F., Zaghouani, W., Widiger, A., Forslund, A., Best, C.: Geocoding multilingual texts: Recognition, Disambiguation and Visualisation. In: Proceedings of LREC 2006, Genoa, Italy, pp. 24–26 (2006)
25. Qi, Y., Candan, K.-S.: CUTS: Curvature-based Development Pattern Analysis and Segmentation for Blogs and Other Text Streams. In: Proceedings of Hypertext 2006, Odense, Denmark (2006)

26. Riloff, E.: Automatically Constructing a Dictionary for Information Extraction Tasks. In: Proceedings of the 11th National Conference on Artificial Intelligence (AAAI 1993). MIT Press, Cambridge (1993)
27. Shannon, C.: A mathematical theory of communication. The Bell System Technical Journal 27 (1948)
28. Tanev, T., Oezden-Wennerberg, P.: Learning to Populate an Ontology of Violent Events. In: Fogelman-Soulie, F., Perrotta, D., Piskorski, J., Steinberger, R. (eds.) NATO Security through Science Series: Mining Massive Datasets for Security. IOS Press, Amsterdam (2008)
29. Tanev, H., Piskorski, J., Atkinson, M.: Real-Time News Event Extraction for Global Crisis Monitoring. In: Kapetanios, E., Sugumaran, V., Spiliopoulou, M. (eds.) NLDB 2008. LNCS, vol. 5039, pp. 207–218. Springer, Heidelberg (2008)
30. Tanev, H., Zavarella, V., Linge, J., Kabadjov, M., Piskorski, J., Atkinson, M., Steinberger, R.: Exploiting Machine Learning Techniques to Build an Event Extraction System for Portuguese and Spanish. LINGUAMÁTICA Journal 2, 55–66 (2009)
31. Wagner, E., Liu, J., Birnbaum, L., Forbus, K., Baker, J.: Using Explicit Semantic Models to Track Situations Across News Articles. In: Proceedings of the AAAI 2006 workshop on Event Extraction and Synthesis, Menlo Park, California, USA (2006)
32. Yangarber, R., Grishman, R.: Machine Learning of Extraction Patterns from Unannotated Corpora. In: Proceedings of the 14th European Conference on Artificial Intelligence: Workshop on Machine Learning for Information Extraction, Berlin, Germany (2000)
33. Yangarber, R.: Counter-Training in Discovery of Semantic Patterns. In: Proceedings of the 41st Annual Meeting of the ACL (2003)
34. Yangarber, R., Von Etter, P., Steinberger, R.: Content Collection and Analysis in the Domain of Epidemiology. In: Proceedings of DrMED 2008: International Workshop on Describing Medical Web Resources at MIE 2008: the 21st International Congress of the European Federation for Medical Informatics 2008, Goeteborg, Sweden (2008)
35. Zavarella, V., Piskorski, J., Tanev, H.: Event Extraction for Italian using a Cascade of Finite-State Grammars. In: Post-Proceedings of the 7th International Workshop on Finite-State Machines and Natural Language Processing, Ispra, Italy (2008/2009)