Real-time News Event Extraction for Global Monitoring Systems

Hristo Tanev, Jakub Piskorski, Martin Atkinson

Joint Research Center of the European Commission
Web and Language Technology Group of IPSC
T.P. 267, Via Fermi 1, 21020 Ispra (VA), Italy
{hristo.tanev,jakub.piskorski,martin.atkinson@jrc.it

Abstract. This paper presents a real-time news event extraction system developed by the Joint Research Centre of the European Commission. It is capable of accurately and efficiently extracting violent and natural disaster events from online news without using much linguistic sophistication. In particular, in our linguistically relatively lightweight approach to event extraction clustered news have been heavily exploited at various stages of processing. The systems' architecture, automatic pattern learning, our new pattern specification language, and information aggregation techniques are briefly described in this paper. Next, the issues of integrating event information in a global monitoring systems are addressed too. Finally, the results of accuracy evaluation are presented.

Key words: event extraction, processing massive datasets, security informatics, machine learning, finite-state technology

1 Introduction

Nowadays, information of any kind is transmitted via Web, mostly in form of a free text. Recently, we have witnessed and ever-growing trend of utilizing natural language processing (NLP) technologies, which go beyond the simple keyword look-up, for automatic knowledge discovery from massive amount of textual data available on the Web.

This paper reports on the event-extraction system developed at the Joint Research Center (JRC) of the European Commission for populating violent incident knowledge base via automatically extracting event information from on-line news articles collected through the Internet with the Europe Media Monitor [1], a web based news aggregation system that receives 40000 news articles from 1400 news sources in 35 languages each day. Gathering information about violent events 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.

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 eventually*

why?. Automatically extracting events is a higher-level information extraction (IE) task which is not trivial due to the complexity of natural language and due to the fact that a full event description is usually scattered over several sentences and documents. Further, event extraction relies on identifying named entities and relations holding among them. Since the latter tasks can be achieved with an accuracy varying from 80 to 90%, obtaining precision/recall figures oscilating around 60% for event extraction (usually involving several entities and relations) is considered to be a good result. The research on automatic event extraction was pushed forward by the DARPA-initiated Message Understanding Conferences (1987-1998) [1], which organized several competitions for the research community in the area of IE, and by the ACE (Automatic Content Extraction) Program [2]. Although, a considerable amount of work on automatic extraction of events have been reported, it still appears to be a lesser studied area in comparison to the somewhat easier tasks of named-entity and relation extraction. Two comprehensive examples of the current functionality and capabilities of event extraction technology dealing with identification of disease outbreaks and conflict incidents are given in [2] and [3] respectively. A most recent trends and developments in this area are reported in [4]

In our linguistically poor approach to event extraction from online news, we take advantage of the fact that news data is clustered. Consequently, only a tiny fraction of each text is analyzed. Further, we deploy simple 1 and 2-slot extracion patterns, which 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 documents voting heuristics are applied in order to aggregate information extracted locally within each cluster. Since efficient processing is a prerequisite for being able to extract event information in real time, we we have developed our own pattern matching engine in order to find a good trade-off between 'compact linguistic descriptions' and efficient processing. The core event extraction system has been intergated with a real-world global monitoring systems. A preliminary evaluation revealed acceptable accuracy and a strong application potential. Although our domain centers around security domain, the techniques deployed in our system can be applied in other domains, e.g., tracking business-related events for risk assessment.

The rest of this paper is organized as follows. First, in section 2 the architecture of our live event extraction processing chain is described. The automatic pattern acquisition technique and our high-speed extraction pattern engine are presented in section 3. Subsequently, section 4 elaborates on information fusion. Section 5 addresses issues concerning integration of the core event extraction system in a global monitoring system and briefly describe visualization aspects. Some evaluation figures are given in 6. Finally, we end up with some conclusions and future directions in section 7.

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

2 Real-time Event Extraction Process

This section briefly describes the real-time event extraction processing chain, which is depicetd in figure 1. First, before the proper event extraction process can proceed, news articles are gathered by dedicated software for electronic media monitoring, namely the EMM system [1], which regularly checks for updates of headline across multiple sites. Secondly, the input data is grouped into news clusters ideally including documents on one topic. Further, clusters describing security-related events are selected via application of key-word based heuristics. For each such cluster the system tries to detect and extract only the main event via analyzing all documents in the cluster.

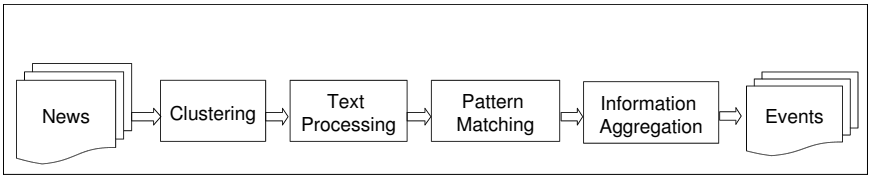

Fig. 1. The architecture of NEXUS

Next, each cluster is processed by NEXUS (News cluster Event eXtraction Using language Structures), our core event extraction engine, which for each detected violent event produces a frame, whose main slots are: date and location, number of killed and injured, kidnapped people, actors, and type of event. NEXUS proceeds as follows. In an initial step, each document in the cluster is linguistically preprocessed in order to produce a more abstract representation of the texts. This encompasses following steps: fine-grained tokenization, sentence splitting, named-entitiy recognition (e.g., people, numbers, locations), simple chunking, labeling of key terms like action words (e.g. *kill*, *shoot*) and unnamed person groups (e.g. *five civilians*). The forementioned tasks are acomplished by CORLEONE (Core Linguistic Entity Online Extraction), our in-house core linguistic engine [5], which is an integral part of NEXUS. In the text preprocessing phase, a geo-coding of documents in each cluster is performed too, which is relevant for defining the place where the main event of the cluster took place.**A REFERENCE TO BRUNOS PAPAER WOULD BE GOOD**

Once texts are grouped into clusters and linguistically preprocessed, the pattern engine applies a cascade of extraction grammars on each document within a cluster. For creating extraction patterns we apply a blend of machine learning and knowledge-based techniques. Firstly, we acquire patterns for recognition of event slots in a semi-automatic manner. Contrary to other approaches, the learning phase is done via exploiting clustered news, which intuitively guarantees better precision of the learned patterns. Secondly, we enhanced the set of automatically learned patterns by adding manually created multi-slot extraction 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.

Finally, since information about events is scattered over different documents, the last step consists of cross-document cluster-level information fusion, i.e., we aggregate and validate information extracted locally from each single document in the same cluster. For this purpose simple voting-like heuristics are deployed, e.g., among the phrases which appear as a filler of a given slot in the event templates extracted from all documents in the cluster, we select the most frequent one.

The output of NEXUS constitutes input for a global monitoring system, addressed in section 5. Noteworthy, the core event-extraction engine is triggered every XXX minutes in order to keep up-to-date. The more thorough description of the automatic pattern acquisition, our pattern engine, and information fusion follows in the subsequent sections. The data gathering and clustering is addressed in [1] and [6].

3 Pattern Learning and Matching

3.1 Pattern Acquisition (HT)

While in the past, IE systems used patterns created manually by human experts, state-of-the-art approaches use machine learning (ML) algorithms for their acquisition [7, 8]. However, ML approaches are never 100% accurate, therefore we manually filter out implausible patterns and add hand-crafted ones, where it is paramount.

Our pattern acquisition approach involves multiple consequtive iterations of ML followed by manual validation. Learning patterns for each event-specific semantic role (e.g. "dead" or "kidnapped") requires a separate cycle of learning iterations. The method uses clusters of news articles produced automatically by our European Media Monitoring Web mining infrastructure [9]. Each cluster includes articles from different sources about the same news story. Therefore, we assume that each entity appears in the same semantic role (actor, victim, injured) in the context of one cluster.

Here are the basic steps of the pattern acquisition algorithm:

1. Annotate a small corpus with event-specific information, e.g., date, place, actors, affected dead, etc. As an example consider the following two sentences:
   a. *<actor>Hezbollah</actor>claimed the responsibility for the kidnapping of the Israeli corporal.*
   b. *<actor>Al Qaida </actor>claimed the responsibility for the bombing which killed <affected_dead>five people</affected_dead>.*

2. Learn automatically single-slot extraction patterns (see....[reference]), e.g., the pattern `[ORGANIZATION] "claimed the responsibility"` could be learned from both sentences, where the entity filling the slot `[ORGANIZATION]` is assigned the role `actor(perpetrator)`
3. Manually check, modify and filter out low quality patterns. Eventually add new patterns. If the size of the list is exceeds certain threshold (the desired coverage is reached)- terminate.

4. Match the patterns against the full corpus or part of it. Next, entities which fill the pattern slots and comply to the semantic constraints of the slot are taken as *anchor entities*. If an anchor entity A (e.g., *five people*) is assigned a role R (e.g., `affected_dead`) in the news cluster C, we assume with high confidence that in the cluster C entity A appears mostly in the same role R. Consequently, annotate automatically all the occurrences of A in C with the label R, e.g., in our example all the occurrences of *five people* in the cluster from which the second sentence originate will be labeled as `affected_dead`.

5. Go to step 2.

After single-slot patterns are acquired by this algorithm, we use some of them to manually create 2-slot patterns like `X shot down Y`. A more cpmrehensive presentation of the learning algorithm is given in [10]

3.2 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 extracion pattern engine [11], 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 [12] and XTDL, a significantly more declarative and linguistically elegant formalism used in a lesser known SProUT platform [13].

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) [3] 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

[3] TFSs are widely used as a data structure for NLP. Their formalizations include multiple inheritance and subtyping, which allow for terser descriptions.

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 the JAPEs feature of associating patterns with multiple actions, i.e., producing multiple annotations (eventually nested) for a given text fragment. Noteworthy, 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: inHowManyEvents(#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 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 `inHowManyEvents()` which contacts some external knowledge base to retrieve the number of events the current actor was involved in the past (such information might be useful in the context of global monitoring systems).

We have compared the run-time behaviour of ExPRESS against the other two pattern engines mentioned earlier. For instance, matching the violenet-event extraction grammar (consisting of ca. 3100 extraction patterns) against various news collection of over 200 MBs can be executed from 12 to 25 times faster than the correspondng XTDL grammar, which have been optimized for speed. A more thorough overview of the techniques for compiling and processing grammars as well as the entire ExPRESS engine is given in [11].

4 Information Aggregation

Our event extraction system firstly uses linear patterns in order to extract entities which have specific semantic roles in each news cluster and secondly merges the single pieces into event descriptions via application of information aggregation algorithm. This algorithm assumes that each cluster reports at most one

main event of interest. It takes on the input the text entities extracted from one news cluster with their semantic roles. The algorithm also considers the sentences from which these entities are extracted.

1. **Disambiguation.** If one and the same entity has two roles assigned, a preference is given to the role assigned by the most reliable group of patterns. The double-slot patterns like `X shot down Y` which extract two entities at the same time are considered the most reliable. Regarding the one-slot constructions, the system considers the ones for detection of `affected_dead`, `affected_wounded`, and `affected_kidnapped` as more reliable than the ones for extraction of the `actor (perpetrator)` (the latter one being more generic). All these preference rules are based on empirical observations.
2. **Counting victims.** Another ambiguity arises from the contradictory information which news sources give about the number of killed, wounded and kidnapped. We use an ad-hoc algorithm for computing the most probable estimation for these numbers. This algorithm finds the biggest group of numbers which are close to each other and then computes their average. After this estimation is computed, the system discards from each news cluster all the articles whose reported victim numbers significantly differ from the estimated numbers for the cluster.
   When victims are counted in each article, a small taxonomy of person classes is used to perform victim arithmetics. For example, if we find in an article that "two soldiers" and "three terrorists" are killed, we sum these numbers, since "soldiers" and "terrorists" belong to different classes. On the other hand, if the article reports about "three gunmen" and "three terrorists", we assume they refer to the same group of people and do not add the numbers, since "gunmen" and "terrorists" belong to the same class of NonGovernmentalArmedGroup
3. **Type definition.** The information aggregation algorithm assigns a class label to the detected violent events where it is possible. Some of the most used event classes are "Terrorist Attack", "Bombing", "Shooting", "Air Attack", etc. The classification algorithm uses a blend of keyword matching and domain specific rules. 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".

The sketched algorithm performs well on our news data. However, it has some limitations, the most important of which is that it considers only one main event per news cluster, ignoring events with smaller importance or incidents subsumed by the main event. In the security related domain it is often necessary to detect links between events. For example, a kidnapping typically includes capturing a hostage, a statement and a video release by the kidnappers in which they declare what they want to liberate the abducted person, police action to liberate the hostage, and finally his or her liberation. The current version of the event extraction algorithm detects these events separately, but it cannot aggregate them into one complex event.

5 Support to Global Crisis Monitoring

The general requirements for Global monitoring emerge from two very different enduser contexts namely situation rooms and actors in the field. The former generally require large displays showing geographical and related territorial information, alerting functionality like audio alarms, zooming to new events, fast indication of the gravity and typology of the event as well as links to relevant information. Actors in the field also have similar functional requirements plus the need to have universal access possibly without the need to download or install dedicated client software applications. A third but nonetheless important end user is a web service that consumes the event meta-data for further integration in other applications.

To fulfill all of these general requirements in one shot we decided that the user interface should be in the form of a Rich Internet Application (RIA)[] with typical Graphical Information System (GIS) features and additionally a standard Desktop Application. For the current demo sites we chose Goggle Maps notably the JavaScript API as the RIA and Google Maps for the desktop application.

First step is to integrate the event meta-data coming from the event extraction system with the source data into a format suitable for transmitting over the internet to the client applications. Currently there are two main formats for transmitting data including the graphical coordinates to client applications that are dedicated or generic web browsers. One is an open format called Geographical Simple Syndication (GeoRSS[]) and the other is a proprietary format Keyhole Markup Language (KML[]). We decided to support both standards in order to support as many of the third user types outlined previously. There are many end clients that support the visualization on maps of GeoRSS data some examples are: Yahoo maps, OpenLayers and WorldKit. Conversely KML[] is mostly supported by the GOOGLE tools [][] however, despite being proprietary it has the advantage of allowing style information to be included in the markup and within the detail of the event. The style information is really important since it allows the data transmitted to include specifications on how the events are to be displayed as icons and the markup of how event details are displayed.

A fourth issue to cover here is due to the fact that the data is visualised by coordinates, a problem occurs when several events are detected at the same location. Unfortunately, at this moment in time, typically. these places are Bagdad and Gaza city. Sometimes the client application handles overlay elegantly (as in Google World), sometimes they do not in which case the overlay has to be dealt with by the server a priori.

The content of the RSS and KML delivered by our system follows as much as possible the standard in terms of tag content. For instance the title is the location, the description has 2 main parts, one is the textual location hierarchy (location, province, region, country), the other is a embedded textual description of all the events at the location.

The textual description of the events contains a list of news clusters of the events at the current location. Each element of this list contains: a title which is the title of the lead cluster which is the tile of the current lead (central) ar-

ticle in the EMM Cluster. The link to the cluster page (on Press.jrc.it), details about the temporal aspects of the clusters story this includes the time the first article was captured, the time of the most recent article in the cluster and the time the event extractor last extracted the meta-data from the cluster. Next is the automatically generated event detail. This a simple phrase built from event attributes as follows (bold indicates the event meta-data) i.e. Event Type: Stabbing. Severity 1 Killed, 0 Injured and 0 Kidnapped. Victims were Rachel Nickell killed and no one injured. Perpetrators were not reported and the weapons used were not reported.. Finally, the description of the main article is included.

To summarize our RIA[] application shows events clustered by location, the icon employed provides 2 visual clues, the first is the main causality which is death followed by injuries followed by kidnapping or undefined as in figure [] the second is the general magnitude in organized into three classes: class 1 is where there are between 1-10 people who are victims, class 2 is where there are 10-100 people as victims and the icon shows an orange perimeter and the final is when there are more than 100 people as victims the icon shows a red halo.

For the WEB based clients that contain the Google Maps, we also implemented a number of additional features, including a Adobe Flash based audio alarm that sounds when the victim count surpasses a user defined threshold. A carousel mode, where all the events are played through in turn are zoomed into the geographical location and the details are popped up in a dialogue. Another feature is the ability to manage the visibility of other layers and allow the free display of other geo-services that provide data in KML or GEORSS.

On the main EMM Site there is also the possibility to see the violent events in a standard (text only view)[].

6 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. Our system detected 27 out of these 29 violent event descriptions (**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%**

Table 1 shows the accuracy of the performance of the victim counting, classification and geo-tagging. The numbers in the table represent the percent of the news clusters for which the corresponding task returned correct result.

The evaluation shows that victims counting and geo-tagging at the level of country nave relatively high performance. Event classification and place-level

Detection task	*Accuracy (%)*
Dead counting	80
Injured counting	93
Event classification	57
Geo-tagging (country)	95
Geo-tagging (place name)	44

Table 1. Event Extraction Performance

geo-tagging can be improved further. The most frequent source of errors for the place-level geo coding is the fact that most of the news agencies are placed in the capital cities, therefore the name of a country capital appears in the beginning of most of the news articles, although the place of the event is different.

7 Conclusions and Future Directions

In this paper, we have presented real time event extraction from on-line news for global crisis monitoring.

In particular, we introduced NEXUS – which performs cluster-level information fusion in order to merge partial information into fully-fledged event descriptions. The results of the preliminary evaluation on violent event extraction show that NEXUS is already operational and can be used for real time global crisis monitoring . Although this first version of our system was profiled to detect mostly violent events and the evaluation was carried out for such kind of events, it turns out that the system detects with high accuracy major man made and natural disasters, which increases its usefulness for crisis monitoring.

All the text processing tools, i.e., CORLEONE, EXPRESS. geo-coding, etc., which we use, are based on finite-state technology in order to fulfill the requirements of a real-time text processing system.

In order to improve the quality of the extracted event descriptions, several system extensions are envisaged. Firstly, some improving the event classification: We are working towards fine grained classification of natural and man made disasters. We also plan to improve the classification accuracy for the violent events. Secondly, we aim at multilingual event extraction and crisis monitoring. This is feasible, since our algorithms and grammars are mostly language independent. Thirdly, 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. Such a structure may provide useful clues for the analysts which study the situation in the World.

8 Acknowledgements

We are indebted to our colleagues without whom the presented work could not have been be possible. In particular, we thank Erik van der Goot, Ralf Steinberger, Bruno Pouliquen, Clive Best, Jenya Belyaeva and other colleagues.

References

1. Best, C., van der Goot, E., Blackler, K., Garcia, T., Horby, D.: Europe Media Monitor. Technical Report EUR 22173 EN, European Commission. (2005)
2. Grishman, R., Huttunen, S., Yangarber, R.: Real-time Event Extraction for Infectious Disease Outbreaks. In Proceedings of Human Language Technology Conference (HLT) 2002, San Diego, USA (2002)
3. 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** (2003) 617–642
4. Ashish, N., Appelt, D., Freitag, D., Zelenko, D.: Proceedings of the workshop on Event Extraction and Synthesis, held in conjcnction with the AAAI 2006 conference. American Association for Artificial Intelligence, Menlo Park, California, USA (2006)
5. Piskorski, J.: CORLEONE – Core Linguistic Entity Online Extraction. Technical report, Joint Research Center of the European Commission, Ispra, Italy (2008)
6. Steinberger, R., Pouliquen, B., Ignat, C.: Navigating multilingual news collections using automatically extracted information. Journal of Computing and Information Technology - CIT **13** (2005) 257–264
7. Jones, R., McCallum, A., Nigam, K., Riloff, E.: Bootstrapping for Text Learning Tasks. In: In Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques, and Applications, Stockholm, Sweden. (1999)
8. Yangarber, R.: Counter-Training in Discovery of Semantic Patterns. In: Proceedings of the 41st Annual Meeting of the Association of Computational Linguistics. (2003)
9. Best, C., Pouliquen, B., Steinberger, R., van der Goot, E., Blackler, K., Fuart, F., Oellinger, T. Ignat, C.: Towards automatic event tracking. In: Intelligence and Security Informatics - Proceedings of IEEE International Conference on Intelligence and Security Informatics (ISI'2006), San Diego, California, USA. (2006) 26–34
10. Tanev, H., Oezden-Wennerberg, P.: Learning to Populate an Ontology of Violent Events (in print). In Perrotta, D. and Piskorski, J. and Soulie-Fogelman, F. and Steinberger, R., ed.: NATO Security through Science Series: Information and Communication Security. IOS Press (2008)
11. 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)
12. Cunningham, H., Maynard, D., Tablan, V.: Jape: a java annotation patterns engine (second edition). Technical Report, CS–00–10, University of Sheffield, Department of Computer Science (2000)
13. Drożdżyński, W., Krieger, H.U., Piskorski, J., Schäfer, U., Xu, F.: Shallow Processing with Unification and Typed Feature Structures — Foundations and Applications. Künstliche Intelligenz **2004(1)** (2004) 17–23