Institute of Informatics
Logics and Security Studies

Bootstrapping Events and Relations from Text

Ting Liu and Tomek Strzalkowski
12/06/2009

ILS Tech Report: [010]

Available for download from:
`http://www.ils.albany.edu`

Abstract

We describe a new approach to semi-supervised adaptive learning of *event extraction* from text. Given a set of examples and an un-annotated text corpus, the BEAR system (Bootstrapping Events And Relations) will automatically learn how to recognize and understand descriptions of complex semantic relationships in text, such as events involving multiple entities and their roles. For example, given a series of descriptions of bombing and shooting incidents (e.g., in newswire) the system will learn to extract, with a high degree of accuracy, other *attack*-type events mentioned elsewhere in text, irrespective of the form of description. A series of evaluations using the ACE test data and event set show a significant performance improvement over previous approaches.

1 Introduction

We constructed a semi-supervised machine learning process that effectively exploits statistical and structural properties of natural language discourse in order to rapidly acquire rules to detect mentions of events and other complex relationships in text, extract their key attributes, and construct template-like representations. The learning process exploits descriptive and structural redundancy, which is common in language; it is often critical for achieving successful communication despite distractions, different contexts, or incompatible semantic models between a speaker/writer and a hearer/reader. We also take advantage of the high degree of referential consistency in discourse (e.g., as observed in word sense distribution by Gale et al, 1992, and arguably applicable to larger linguistic units), which enables the reader to efficiently correlate different forms of description across coherent spans of text.

The method we describe here consists of two steps: (1) supervised acquisition of initial extraction rules from an annotated training corpus, and (2) self-adapting unsupervised multi-pass bootstrapping by which the system learns new rules as it reads un-annotated text using the rules learnt in the first step and in the subsequent learning passes. When a sufficient quantity and quality of text material is supplied, the system will learn many ways in which a specific class of events can be described. This includes the capability to detect individual event mentions using a system of context-sensitive triggers and to isolate pertinent attributes such as agent, object, instrument, time, place, etc., as may be specific for each type of event. This method produces an accurate and highly adaptable event extraction that significantly outperforms current information extraction techniques both in terms of accuracy and robustness, as well as in deployment cost.

2 Learning by bootstrapping

As a semi-supervised machine learning method, bootstrapping can start either with a set of predefined rules or patterns, or with a collection of training examples (seeds) annotated by a domain expert on a (small) data set. These are normally related to a target application domain and may be regarded as initial

"teacher instructions" to the learning system. The training set enables the system to derive initial extraction rules, which are applied to un-annotated text data in order to produce a much larger set of examples. The examples found by the initial rules will oc cur in a variety of linguistic contexts, and some of these contexts may provide support for creating *alternative extraction rules*. When the new rules are subsequently applied to the text corpus, additional instances of the target concepts will be identified, some of which will be positive and some not. As this process continues to iterate over, the system acquires more extraction rules, fanning out from the seed set until no new rules can be learned.

Thus defined, bootstrapping has been used in natural language processing research, notably in word sense disambiguation (Yarowsky, 1995), for named entity extraction (Strzalkowski and Wang, 1996), and more recently for simple relation extraction (Agichtein and Gravano, 2000; Yangarber, 2000). Bootstrapping is an attractive learning technique because it partly overcomes the problem of manual design as well as costly retraining and tuning required in supervised machine learning methods. However, implementing a successful bootstrapping system is a difficult task and the technique has thus been only applied to relatively structurally simple problems, as noted above. It has not yet been successfully applied to learning of complex relations such as event extraction.

Strzalkowski and Wang (1996) were first to demonstrate that the technique could be applied to adaptive learning of named entity extraction rules. For example, given a "naïve" rule for identifying company names in text, e.g., "capitalized NP followed by *Co.*", their system would first find a large number of (mostly) positive instances of company names, such as "*Henry Kauffman Co.*" From the context surrounding each of these instances it would isolate alternative indicators, such as "*the president of*", which is noted to occur in front of many company names, as in "*The president of American Electric Automobile Co*. …". Such alternative indicators give rise to new extraction rules, e.g., "*president of* + CNAME". The new rules find more entities, including company names that do not end with *Co.*, and the process iterates until no further rules are found. The technique achieved a very high performance (95% precision and 90% recall), which encouraged more research in IE area by using bootstrapping techniques. In Snowball, (Agichtein and Gravano 2000) applied bootstrapping technique to extraction of binary relations, such as Organization-Location, e.g., between *Microsoft* and *Redmond, WA*. Yangarber (2000) adapted this method to documents classification by learning to correlate the presence of certain subject-verb-object patterns in text with the document topic.

3 Bootstrapping applied to event learning

Our objective in this project was to expand the bootstrapping technique to learn extraction of complex relations from text, irrespective of their form of description, a property essential for successful adaptability to new domains and text genres. The major challenge in advancing from entities and binary relations to event learning is the complexity of structures involved that not only consist of multiple elements but their linguistic context may now extend well beyond a few surrounding words, even past sentence boundaries. These considerations guided

the design of the BEAR system (Bootstrapping Events And Relations) which is described in this paper.

3.1 Event representation

An event description can vary from very concise, newswire-style to very rich and complex as may be found in essays and other narrative forms. The system needs to recognize any of these forms and to do so we need to distill each description to a basic event pattern. This pattern will capture the heads of key phrases and their dependency structure while suppressing modifiers and certain other non-essential elements. Such skeletal representations cannot be obtained with keyword analysis or linear processing of sentences at word level (e.g., Agichtein and Gravano 2000; Brin 1998), because such methods cannot distinguish a phrase head from its modifier. On the other hand, a full syntactic parser is not required for the task, and becomes problematic when large datasets need to be processed due to increased computational cost (e.g., Stanford parser; Klein and Manning, 2003). A shallow dependency parser, such as Minipar (Lin 1998), that recognizes dependency relations between words is quite sufficient for deriving head-modifier relations and thus for construction of event templates. Event templates are obtained by stripping the parse tree of modifiers while preserving the basic dependency structure as shown in Figure 1, which is a stripped down parse tree of, "*Also Monday, Israeli soldiers **fired** on four diplomatic vehicles in the northern Gaza town of Beit Hanoun, said diplomats*"

The model proposed here represents a significant advance over the current methods for relation extraction, such as the SVO model (subject- verb-object triples, e.g., Yangarber et al., 2000) and its extension, e.g., the chain model (Sudo 2001) and other related variants (Riloff, 1996) all of which are too simple to accurately recognize and represent complex event descriptions to support successful machine learning. While Sudo's subtree model (2003) overcomes some of the limitations of the chain models and is thus conceptually closer to our method, it nonetheless lacks efficiency required for practical applications.

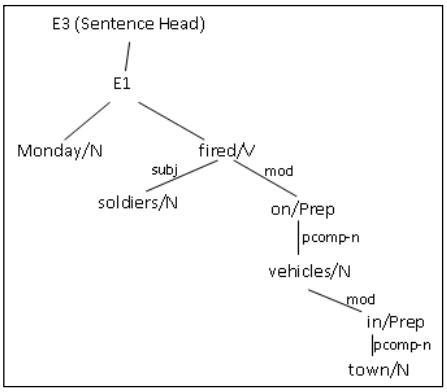

Figure 1. Skeletal dependency structure representation of an event mention.

We represent complex relations as tree-like structures anchored at an *event trigger* (which is usually but not necessarily the main verb) with branches extending to the event attributes (which are usually named entities). Unlike the singular concepts (i.e., named entities such as 'person' or 'location') or linear relations (i.e., tuples such as 'Gates – CEO – Microsoft'), an event description consists of elements that form non-linear dependencies, which may not be

apparent in the word order and therefore require syntactic and semantic analysis to extract. Furthermore, an arrangement of these elements in text can vary greatly from one event mention to the next, and there is usually other intervening material involved. Consequently, we construe event representation as a collection of *paths* linking the trigger to the attributes through the nodes of a parse tree[1].

To create an event pattern (which will be part of an extraction rule), we generalize the dependency paths that connect the event trigger with each of the event key attributes (the *roles*). A dependency path consists of lexical and syntactic relations (POS and phrase dependencies), as well as semantic relations, such as entity tags (e.g., Person, Company, etc.) of event roles and word sense designations (based on Wordnet senses) of event triggers. In addition to the trigger-role paths (which we shall call the *sub-patterns*), an event pattern also contains the following:

- Event Type and Subtype – which is inherited from seed examples;
- Trigger class – an instance of the trigger must be found in text before any patterns are applied;
- Confidence score – expected accuracy of the pattern established during training process;
- Context profile – additional features collected from the context surrounding the event description, including references of other types of events near this event, in the same sentence, same paragraph, or adjacent paragraphs.

We note that the trigger-attribute sub-patterns are defined over phrase structures rather than over linear text, as shown in Figure 2. In order to compose a complete event pattern, sub-patterns are collected across multiple mentions of the same-type event.

Attacker: <N = Attacker(subj, PER)> <V = trigger>
Place: <V = trigger> <Prep> <N> <Prep = in> <N = Place(GPE)>
Target: <V = trigger> <Prep = on> <N = Target(VEH)>
Time-Within:<N = Time-Within(timex2)><SentHead><V = trigger>

Figure 2. Trigger-attribute sub-patterns for key roles in a *Conflict-Attack* event pattern.

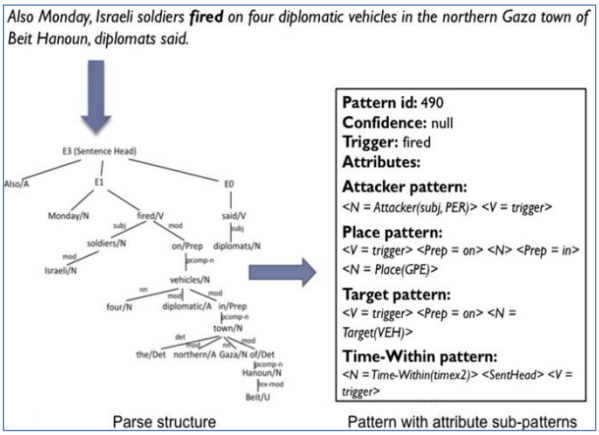

Figure 3. A *Conflict-Attack* event pattern derived from a positive example in the training corpus

[1] Details of how to derive the skeletal tree representation are described in (Anonymous, 2009).

3.2 Deriving initial rules from seed examples

Extraction rules are construed as transformations from the event patterns derived from text onto a formal representation of an event.

The initial rules are derived from a manually annotated training text corpus (seed data), supplied as part of an application task. Each rule contains the type of events it extracts, trigger, a list of role sub-patterns, and the confidence score obtained through a validation process (see section 3.5).

Figure 3 shows an extraction pattern for the Conflict-Attack event derived from the training corpus (but not validated yet)[2].

As an alternative to using seed data, we are also exploring an option of utilizing an existing event extraction system from a related domain (or general domain) to seed the application data.

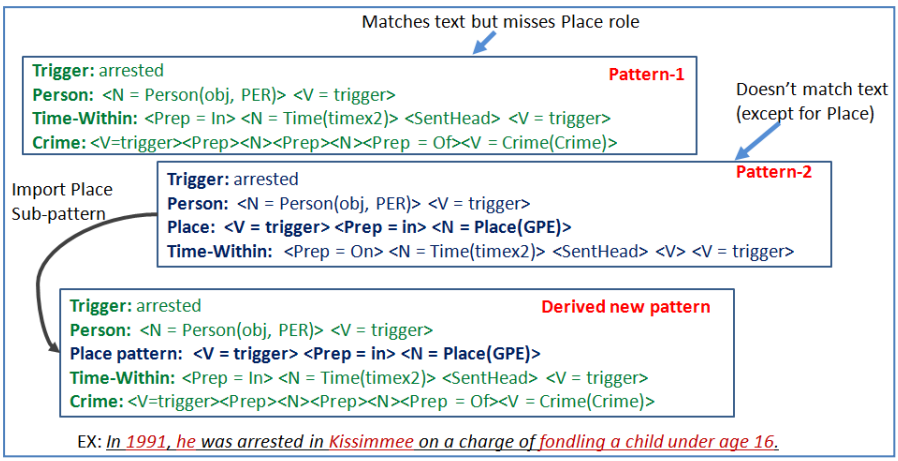

Figure 4. Deriving a new pattern by importing a role from

3.3 Learning through pattern mutation

Given an initial set of extraction rules, a variety of pattern mutation techniques are applied to derive new patterns and new rules. This is done by selecting elements of previously learnt patterns, based on the history of partial matches and combining them into new patterns. This form of learning, which also includes conditional rule relaxation, is particularly useful for rapid adaptation of extraction capability to slightly altered, partly ungrammatical, or otherwise variant data. The basic idea is as follows: the patterns acquired in prior learning iterations (starting with those obtained from the seed examples) are matched against incoming text to extract new events. Along the way there will be a number of partial matches, i.e., when no existing pattern fully matches a span of text. This may simply mean that no event is present; however, depending upon the degree of the partial match we may also consider that a novel structural variant was found. BEAR would automatically test this hypothesis by attempting to construe a new pattern, out of the elements of existing patterns, in order to achieve a full match. If a match is achieved, the new "mutated" pattern will be added to BEAR learned collection, subject to a validation step. The validation step (discussed later in this paper) is to assure that the added pattern would not

[2] In this figure we omit the parse tree trimming step which was explained in the previous section.

introduce an unacceptable drop in overall system precision. Specific pattern mutation techniques include the following:

- Adding a role subpattern: When a pattern matches an event mention while there is sufficient linguistic evidence (e.g., presence of certain types of named entities) that additional roles may be present in text, then appropriate role subpatterns can be "imported" from other, non-matching patterns (Figure 4).
- Replacing a role subpattern: When a pattern matches but for one role, the system can replace this role subpattern by another subpattern for the same role taken from a different pattern for the same event type.
- Adding or replacing a trigger: When a pattern matches but for the trigger, a new trigger can be added if it is already present in another pattern for the same event type.

We should point out that some of the same effects can be obtained by making patterns more general, i.e., adding "optional" attributes, etc. Nonetheless, the pattern mutation will automatically learn such generalization on as-needed basis in an entirely data-driven fashion, while also maintaining high precision of the resulting patterns. It is thus a more general method. Figure 4 illustrated the use of the elements combination technique. In this example, neither of the two existing patterns can fully match the new event description; however, by combining the first pattern with the Place role sub-pattern from the second pattern we obtain a new pattern that fully matches the text. While this adjustment is quite simple, it is nonetheless performed automatically and without any human assistance. The new pattern is then "learned" by BEAR, subject to verification step explained in a later section.

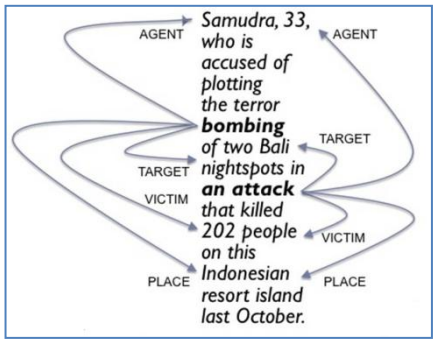

Figure 5. A new extraction pattern is derived by identifying an alternative trigger

3.4 Learning by exploiting structural duality

As the system reads through new text extracting more events using already learnt rules, each extracted event mention is analyzed for presence of alternative trigger elements that can consistently predict the presence of a subset of events that includes the current one. Subsequently, an alternative sub-pattern structure will be built with branches extending from the new trigger to the already identified attributes, as shown schematically in Figure 5. In this example, a *Conflict-Attack*-type event is extracted using a pattern (shown in Figure 5A) anchored at the "*bombing*" trigger. Nonetheless, an alternative trigger structure is discovered, which is anchored at "an attack" NP, as shown on the right side of Figure 5. This "discovery" is based upon seeing the new trigger repeatedly – it needs to "explain" a subset of previously seen events to be adopted. The new

trigger will prompt BEAR to derive additional event patterns, subject to confidence validation, including the one in Figure 5B. The resulting extraction rules will be immediately applied to extract more events.

PatternID: 1207
Type: Conflict **Subtype:** Attack
Trigger: bombing
Target: <N = trigger> <Prep = of> <N = Target(FAC)>
Attacker: <N = Attacker(ORG)><V> <N> <Prep = for> <N = trigger>
Time-Within: <N = trigger> <Prep> <N> <Prep> <N> <V> <N = Time-within(timex2)>

Figure 5A. A pattern with the *bombing* trigger matches the event mention in Fig. 5.

PatternID: 1286
Type: Conflict **Subtype:** Attack
Trigger: attack
Target: <N = Target(FAC)> <Prep = in> <N = trigger>
Attacker: <N = Attacker(ORG)> <V> <N> <Prep> <N> <Prep = in> <N = trigger>
Time-Within: <N = trigger> <V> <N = Time-within(timex2)>

Figure 5B. A new pattern is derived for event in Fig 5, with an *attack* as the

Another way of getting at this kind of structural duality is to exploit co-referential consistency within coherent spans of discourse, e.g., a single news article or a similar document. Such documents may contain references to multiple events, but when the same type of event is mentioned along with the same attributes, it is more likely than not in reference to the same event.

This hypothesis is based on an argument advanced in (Gale et al., 1992) that a polysemous word used multiple times within a single document, is consistently used in the same sense. While that work focused on lexical coherence, a broader principle of referential coherence may apply to more complex structures, such as maintaining coherent references to entities and events within the same discourse to make it understandable. If this principle holds, then multiple mentions of an event within a document and involving many similar elements (type, attributes) must indeed be considered co-referential.

Indeed, Ji (2008) showed that trigger co-occurrence helps finding new mentions of the same event; however, it alone does not guarantee a high precision of detecting coreference. Our experiments (conducted on ACE training corpus) showed that the likelihood of in-document event mentions being co-referential is less than 55% if based just on the co-occurrence of the same trigger, but it can goes to more than 95% when the common trigger has a fare confidence score (0.5 or more) and there are at least 2 shared roles identified[3]. Thus our approach uses co-occurrence of both trigger and event argument for detecting coreferential mentions.

[3] An additional condition here would be that there are no overt role conflicts between the event mentions.

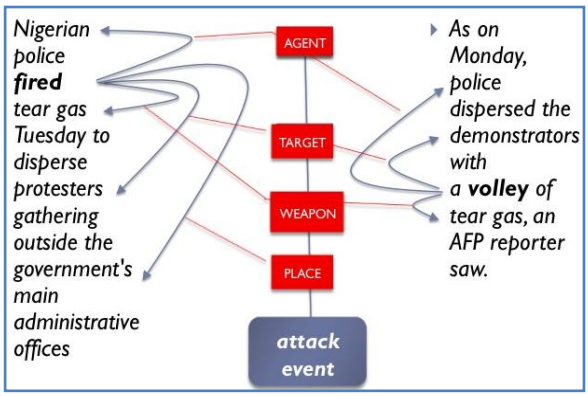

Figure 6. Two event mentions have different triggers and sub-patterns

Furthermore, as many of these co-referential mentions use alternative syntactic and semantic structures, they can be exploited to acquire still more extraction patterns. This principle is illustrated schematically in Figure 6.
In this example, two separate mentions of a *Conflict-Attack* event are found in the same document, but only one of them (the one on the left) is extracted with an existing pattern. The second mention is not detected at first but its parse structure reveals a number of common features with the first mention. For example, both mentions have several entities in common, including at least two roles (*police* and *tear gas*) and moreover we also determine (via Wordnet) that "*volley*" is the hyponym of "*fire*", which is the trigger in the first sentence. This constitutes sufficient evidence to conclude that the two mentions are co-referential[4]; BEAR thus learns a new pattern for *Conflict-Attack*, which is obtained by linking "*volley*" as a new trigger to the already identified roles, as shown in Figure 6A.

PatternID: 257
Type: Conflict
Subtype: Attack
Trigger: volley
Attacker: <N=Attacker(PER)> <V> <N> <Prep=with> <N=trigger>
Instrument: <N=trigger> <Prep=of> <N=Instrument(WEA)>

Figure 6A. A new pattern for *Conflict-Attack* learned by exploiting event co-reference.

3.5 Pattern validation

Extraction patterns are validated after each learning cycle against the already annotated data. In the first supervised learning step, patterns accuracy is tested against the training corpus; in the subsequent steps the validation is extended over parts of the test corpus, so that the annotations obtained with the previously learnt rules can be used to validate new rules. This is accomplished by counting the rate at which each new rule "rediscovers" already extracted events of the same type vs. picking up events of a wrong type. Event patterns are thus validated for overall expected precision by calculating the ratio of positive matches to all matches against known events. This produces pattern confidence scores, which are used to decide if a pattern is to be learned or not. Learning only the patterns with sufficiently high confidence scores helps to guard the bootstrapping process from spinning off track; nonetheless, the overall objective

[4] Note that there is no conflict on TIME attribute: the first mention happened on Tuesday, the second on a day other than Monday.

is to maximize the performance of the resulting set of extraction rules, particularly by expanding its recall rate.

In addition to the confidence rate of each new pattern, we also calculate projected accuracy of each of the role sub-patterns and the triggers, because they may be used in the process of detecting new patterns, and it will be necessary to score partial matches, as a function confidence weights for pattern components. We estimate selectivity of each trigger (i.e., how well it identifies the correct event type), precision of role sub-patterns, as well as impact of certain other contextual features such as mentions of other events nearby (e.g., a mention of Die-type event in the same paragraph increases likelihood of an Attack event, etc.). Similarly, some role sub-patterns (such as <Prep=*with*><NP=weapon>) are highly predictive of specific event types (e.g., *Conflict-Attack*) and their presence may trump a low selectivity trigger (e.g., *hit*) prompting BEAR to assign high confidence score to the entire pattern.

3.6 Recognizing event coreference

While the techniques described above help BEAR learn patterns that will extract more and different mentions of events, the system is also required to recognize co-referential descriptions, i.e., when the same event is mentioned in different documents and sources (in addition to in-document co-reference). For example, the following sentences refer to the same event using radically different descriptions:

"*Georgia's president said Russian aircraft* ***bombed*** *several Georgian villages and other civilian facilities.*"

"*A senior Russian diplomat in charge of the South Ossetian conflict, Yuri Popov, dismissed the Georgian claims of Russian* ***bombings*** *as misinformation, the RIA-Novosti news agency reported.*"

In order to complete the event extraction process BEAR needs to identify which of the extracted event mentions are co-referential – this is the basis for evaluating extraction accuracy in ACE, which we also use to measure the performance of BEAR.

Accordingly, we have implemented an approach to event co-reference based on a set of contextual features that include the type of event, correlation of the trigger elements (synonyms, variants, etc. e.g., via Wordnet), presence of demonstrative pronouns (this, that, etc.), presence of shared attributes in shared roles, among others.

Our approach is quite similar to the techniques used by Grishman *et al*. (2005) and Ahn (2006); however, we avoid some of their performance issues by collecting more features on which to base co-reference decisions and then carefully balancing the impact of general and specific features. In particular we perform more context checks, e.g., to make sure that the same entity is indeed playing the same role in various event mentions (i.e., not to confuse Oswald shooting Kennedy with Oswald being shot himself). We also bring the larger context to bear by considering each event mention as part of a larger "scenario" and noting if other elements of this scenario appear in an immediate context of each mention – which includes preceding and following sentences as well as other events in them[5].

[5] Space limitations do not allow for a full presentation of our co-reference approach and it will be described in a future paper.

4 Evaluation

In this section, we discuss the experiments conducted to evaluate the performance of the techniques underlying BEAR: how effectively it can learn and how accurately it can perform its extraction task. We test the system learning effectiveness by comparing its performance immediately following the first iteration (i.e., using rules derived from the training data) with its performance after N cycles of unsupervised learning.

We run experiments evaluating different dimensions of system performance. First, we evaluated BEAR performance at the task of extracting event mentions from text, as well as the effectiveness of different pattern learning methods. Next we evaluated BEAR on full event extraction compared its performance against other systems participating in ACE 2005[6]. Table 1 shows the types of events defined in ACE 2005.

Types	Subtype
Life	Be-Born, Marry, Divorce, Injure, Die
Movement	Transport
Transaction	Transfer-Ownership, Transfer-Money
Business	Start-Org, Merge-Org, Declare-Bankruptcy, End-Org
Conflict	Attack, Demonstrate
Contact	Meet, Phone-Write
Personnel	Start-Position, End-Position, Nominate, Elect
Justice	Arrest-Jail, Release-Parole, Trial-Hearing, Charge-Indict, Sue, Convict, Sentence, Fine, Execute, Extradite, Acquit, Appeal, Pardon

Table 1. Event types and subtypes in ACE 2005.

4.1 Evaluating learning effectiveness

During each learning cycle, BEAR calculates the confidence scores of the new patterns (i.e., their projected accuracy), including the sub-patterns, and triggers discovered in this iteration, and it learns them only if a confidence score is high enough, i.e., it exceeds a preset threshold. This learning threshold can be adjusted to allow

BEAR learn more patterns at the expense of decreased precision. If an application requires more precision this threshold should be set higher; if more recall is needed it can be set lower.

[6] We used ACE training data (599 documents from news, weblog, usenet, and conversational telephone speech) as training set and evaluate on ACE test data (155 documents).

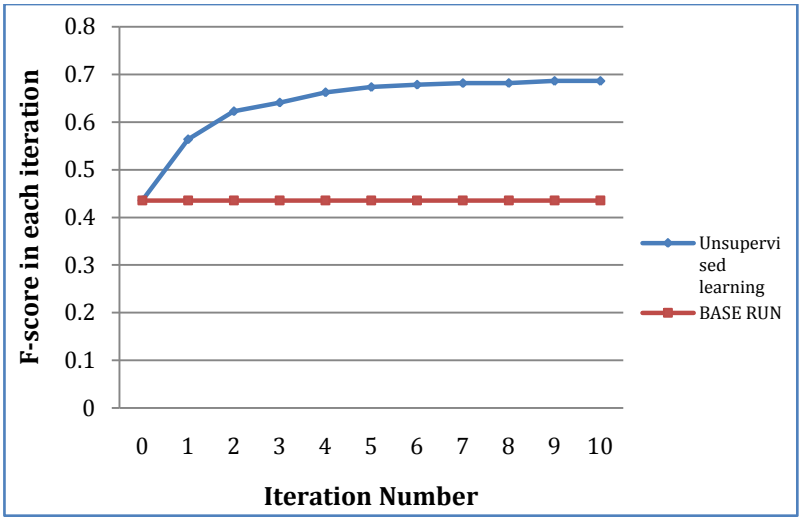

Figure 7. BEAR's unsupervised learning

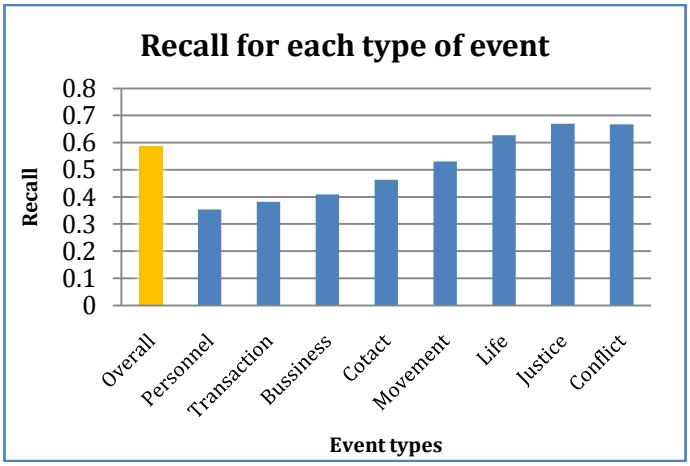

Figure 8. Event mention extraction after learning: recall for each type of event

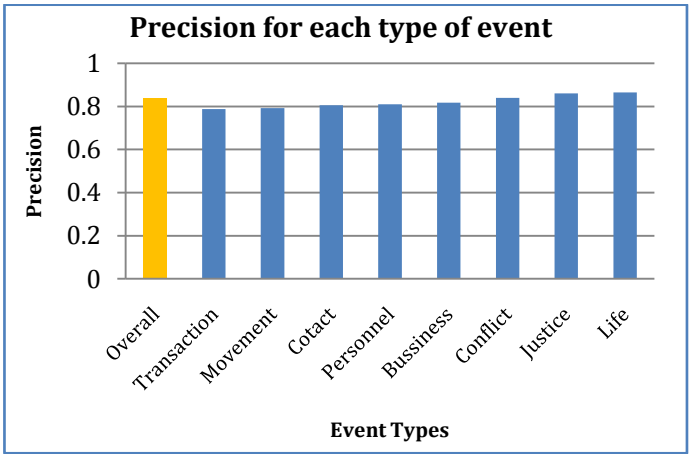

Figure 9. Event mention extraction after learning: precision for each type of event

Figure 7 explains BEAR's learning effectiveness at what we determined empirically to be the optimal confidence threshold for pattern acquisition. We

note that the performance of the system steadily increases until it reaches a plateau after about 10 learning cycles.

The next two charts show a detailed breakdown of BEAR extraction performance after 10 learning cycles for different types of events. We note that while precision holds steady across the event types, recall levels vary significantly. The main reason for low recall in some types of events is the failure to find a sufficient number of high-confidence patterns. This may point to limitations of the current pattern discovery methods and may require new ways of reaching outside of the current feature set.

In the previous section we described several learning methods that BEAR uses to discover, validate and adapt new event extraction rules. Some of them work by manipulating already learnt patterns and adapting them to new data in order to create new patterns, and we shall call these *pattern-mutation methods* (PMM). Other described methods work by exploiting a broader linguistic context in which the events occur, or *context-based methods* (CBM). CB methods look for structural duality in text surrounding the events and thus discover alternative extraction patterns.

	P	R	F	A	T	C	FA
Base	0.89	0.29	0.44	1390	447	400	47
All	0.82	0.59	0.69	1390	1008	823	185
PMM	0.83	0.52	0.64	1390	863	719	144
CBM	0.84	0.44	0.58	1390	729	614	115

Table 2. BEAR performance following different selections of learning steps.[7]

In Table 2, we report the results of running BEAR with each of these two groups of learning methods separately and then in combination to see how they contribute to the end performance. We show BEAR performance after 10 learning iterations operating at the optimal learning threshold. As explained before, new extraction rules are learned in each iteration cycle, based on what was learned in prior cycles and that new rules are adopted only after they are tested for their projected accuracy (confidence score), so that the overall precision of the resulting rule set is maintained at a high level relative to the base run.

When we examine the results of the final run, we notice that the pattern-based methods and context-based methods learn different extraction rules, which in turn extract different, though overlapping, sets of events. The combined set of methods is significantly more effective than either of the two groups taken separately. With the combined methods the system learns enough to deliver significantly improved extraction performance with the F-score increasing by more than 20% over the base run.

[7] What the columns mean in Table 2: P – Precision, R – Recall, F – F-score, A – # of annotated events, T – # of events extracted, C – # correctly extracted, FA – # false alarms.

4.2 Evaluating event extraction

The BEAR system was also tested against ACE event extraction data from ACE 2005 diagnostic competitions[8] because we were interested how its performance compares to that of other leading event extraction systems, on a common task. The objective of this test was to extract and classify all mentions of 33 event types from a text corpus, and then combine co-referential mentions into complete event representations. In order to compare the results we score BEAR runs with ACE Value metric, which combines credits for complete and partial matches against the human prepared answer key with penalties for false alarms and wrongly matched events(ACE, 2005). Table 3 shows that the bootstrapping process helps BEAR to learn rapidly and that the resulting system outperforms the best ACE results.

Systems	Value score
Bear after learning	34.4
BBN	32.7
NYU	29.9
BEAR base run	20.9
University of Amsterdam	19.7

Table 3. BEAR compared with other ACE systems

5 Conclusion and future work

In this paper, we presented a semi-supervised method for learning new event extraction patterns from un-annotated text. The techniques described here add significant new tools that increase capabilities of information extraction technology in general, and more specifically, of systems that are built by purely supervised methods or from manually designed rules. We also described an effective method for finding co-referential events within the same document. Our evaluation using ACE dataset demonstrated that that bootstrapping can be effectively applied to learning event extraction rules and that the resulting system can outperform supervised or manually constructed systems.

Some follow-up research issues include:

- New techniques are needed to recognize event descriptions that still evade the current pattern derivation techniques, especially for the events defined in *Personnel*, *Business*, and *Transactions* classes.
- Adapting the bootstrapping method to extract events in different language, e.g. Chinese or Arabic.
- Expanding this method to extraction of larger "scenarios", i.e., groups of correlated events that form coherent "stories" often described in larger sections of text, e.g., an event and its immediate consequences.

[8] The diagnostic task provides annotated entity/time/value in both training and test data.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting Relations from Large Plain-Text Collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*

David Ahn. 2006. The Stages of Event Extraction. In *Proceedings of the Workshop on Annotating and Reasoning About Time and Events*

Sergey Brin. 1998. Extracting patterns and relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*.

William A. Gale, Kenneth W. Church, and David Yarowsky. 1992. One sense per discourse. In *Proceedings of the workshop on Speech and Natural Language*, 233-237. Harriman, New York: Association for Computational Linguistics.

Ralph Grishman and Beth Sundheim. 1995. Design of the MUC-6 evaluation. 1-11, Columbia, Maryland: Association for Computational Linguistics

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 System Description. In *2005 ACE Evaluation*.

Heng Ji and Ralph Grishman. 2008. Refining Event Extraction Through Unsupervised Cross-document Inference. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL 2008)*.Ohio, USA.

Dan Klein and Christopher D. Manning. 2003. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Proceedings of Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Cambridge, MA: MIT Press, pp. 3-10.

George R. Krupka. 1995. SRA: description of the SRA system as used for MUC-6. In *Proceedings of the 6th conference on Message understanding*, 221-235, Columbia, Maryland: Association for Computational Linguistics

Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing System,* Granada, Spain.

Ting Liu. 2009. BEAR: Bootstrap Event and Relations from Text. Thesis

Ellen Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049. The AAAI Press/MIT Press.

Kiyoshi Sudo, Satoshi Sekine, Ralph Grishman. 2001. Automatic Pattern Acquisition for Japanese Information Extraction. In *Proceedings of Human Language Technology Conference (HLT2001)*.

Kiyoshi Sudo, Satoshi Sekine, Ralph Grishman. 2003. An improved extraction pattern representation model for automatic IE pattern acquisition. *Proceedings of ACL 2003* , 224 – 231. Tokyo.

The ACE 2005 (ACE05) Evaluation Plan. http://www.nist.gov/speech/tests/ace/2005/doc/ace05-evalplan.v3.pdf

Tomek Strzalkowski and Jin Wang. 1996. A self-learning universal concept spotter. In *Proceedings of the 16th conference on Computational linguistics - Volume 2*, 931-936, Copenhagen, Denmark: Association for Computational Linguistics

Carl Weir, Tim Finin, Robin McEntire, and Barry Silk. 1991. Unisys: description of the Unisys system used for MUC-3. 212-222, San Diego, California: Association for Computational Linguistics

Ralph Weischedel, Damaris Ayuso, Sean Boisen, Robert Ingria, and Jeff Palmucci. 1991. BBN: description of the PLUM system as used for MUC-3. In *Proceedings of the 3rd conference on Message understanding*, 137-143, San Diego, California: Association for Computational Linguistics

Ralph Weischedel, *et al.* 1993. BBN: description of the PLUM system as used for MUC-5. In *Proceedings of the 5th conference on Message understanding*, 93-107, Baltimore, Maryland: Association for Computational Linguistics

Ralph Weischedel. 1995. BEN: description of the PLUM system as used for MUC-6. In *Proceedings of the 6th conference on Message understanding*, 55-69, Columbia, Maryland: Association for Computational Linguistics

Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. 2000. Unsupervised discovery of scenario-level patterns for information extraction. In *Proceedings of the Sixth Conference on Applied Natural Language Processing, (ANLP-NAACL 2000)*, 282-289

David Yarowsky. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, 189-196, Cambridge, Massachusetts: Association for Computational Linguistics