Bootstrapping Events and Relations from Text

Ting Liu
ILS, University at Albany, USA
tliu@albany.edu

Tomek Strzalkowski
ILS, University at Albany, USA
Polish Academy of Sciences
tomek@albany.edu

Abstract

In this paper, 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 data and event set show a significant performance improvement over our baseline system.

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 occur 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). 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. Using a similar approach, (Thelen and Riloff, 2002) generated new syntactic patterns by exploiting the context of known seeds for learning semantic categories.

In Snowball (Agichtein and Gravano, 2000) and Yangarber's IE system (2000), bootstrapping technique was applied for extraction of binary relations, such as Organization-Location, e.g., between *Microsoft* and *Redmond, WA*. Then, Xu (2007) extended the method for more complex relations extraction by using sentence syntactic structure and a data driven pattern generation. In this paper, we describe a different approach on building event patterns and adapting to the different structures of unseen events.

3 Bootstrapping applied to event learning

Our objective in this project was to expand the bootstrapping technique to learn extraction of events 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), because such methods cannot distinguish a phrase head from its modifier. 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 (Yangarber, et al. 2000) and its extension, e.g., the chain model (Sudo, et al. 2001) and other related variants (Riloff, 1996) all of which lack the expressive power to accurately recognize and represent complex event descriptions and 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.

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

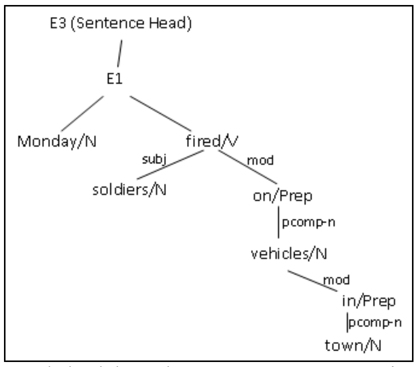


Figure 1. Skeletal dependency structure representation of an event mention.

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

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

3.2 Designating the sense of event triggers

An event trigger may have multiple senses but only one of them is for the event representation. If the correct sense can be determined, we would be able to use its synonyms and hyponym as alternative event triggers, thus enabling extraction of more events. This, in turn, requires sense disambiguation to be performed on the event triggers.

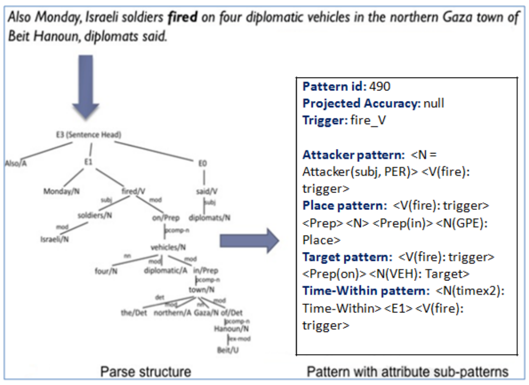
In MUC evaluations, participating groups (Yangarber and Grishman, 1998) used human experts to decide the correct sense of event triggers and then manually added correct synonyms to generalize event patterns. Although accurate, the process is time consuming and not portable to new domains.

We developed a new approach for utilizing Wordnet to decide the correct sense of an event trigger. The method is based on the hypothesis that event triggers will share same sense when represent same type of event. For example, when the verbs, *attack*, *assail*, *strike*, *gas*, *bomb*, are trigger words of Conflict-Attack event, they share same sense. This process is described in the following steps:

1) From training corpus, collect all triggers, which specify the lemma, POS tag, the type of event and get all possible senses of them from Wordnet.
2) Order the triggers by the *trigger frequency* TrF(t, w_pos),[2] which is calculated by dividing number of times each word (w_pos) is used as a trigger for the event of type (t) by the total number of times this word occurs in the training corpus. Clearly, the greater trigger frequency of a word, the more discriminative it is as a trigger for the given type of event. When the senses of the triggers with high accuracy are defined, they can be the reference for the triggers in low accuracy.
3) From the top of the trigger list, select the first none-sense defined trigger (Tr1)
4) Again, beginning from the top of the trigger list, for every trigger Tr2 (other than Tr1), we look for a pair of compatible senses between Tr1 and Tr2. To do so, traverse Synonym, Hypernym, and Hyponym links starting from the sense(s) of Tr2 (use either the sense already assigned to Tr2 if has or all its possible senses) and see whether there are paths which can reach the senses of Tr1. If such converging paths exist, the compatible senses

[1] Details of how to derive the skeletal tree representation are described in (Liu, 2009).
[2] t – the type of the event, w_pos – the lemma of a word and its POS.
[3] In this figure we omit the parse tree trimming step which was explained in the previous section.

[2] t – the type of the event, w_pos – the lemma of a word and its POS.



Pattern id: 490
Projected Accuracy: null
Trigger: fire_V

Attacker pattern: <N = Attacker(subj, PER)> <V(fire): trigger>
Place pattern: <V(fire): trigger> <Prep> <N> <Prep(in)> <N(GPE): Place>
Target pattern: <V(fire): trigger> <Prep(on)> <N(VEH): Target>
Time-Within pattern: <N(timex2): Time-Within> <E1> <V(fire): trigger>

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

are identified and assigned to Tr1 and Tr2 (if Tr2's sense wasn't assigned before). Then go back to step 3. However, if no such path exist between Tr1 senses with other triggers senses, the first sense listed in Wordnet will be assigned to Tr1

This algorithm tries to assign the most proper sense to every trigger for one type of event. For example, the sense of *fire* as trigger of Conflict-Attack event is "start firing a weapon"; while it is used in Personal-End_Position, its sense is "terminate the employment of". After the trigger sense is defined, we can expand event triggers by adding their synonyms and hyponyms during the event extraction.

3.3 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 3.6). Figure 3 shows an extraction pattern for the Conflict-Attack event derived from the training corpus (but not validated yet)[3].

3.4 Learning through pattern mutation

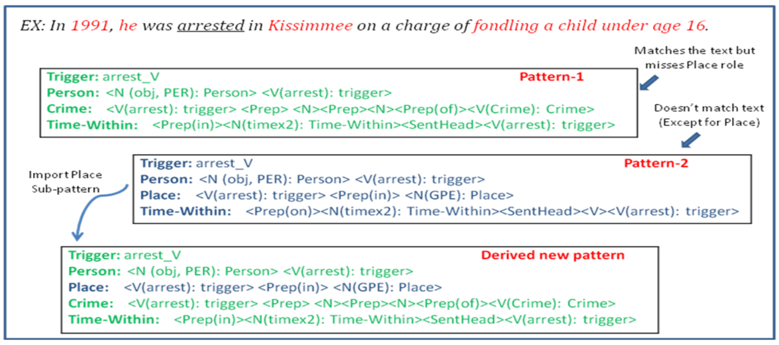
Given an initial set of extraction rules, a variety of pattern mutation techniques are applied to derive new patterns and new rules. This is done by selecting elements of previously learnt patterns, based on the history of partial matches and combining them into new patterns. This form of learning, which 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 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 a 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 either is already present in another pattern for the same event type or the synonym/hyponym/hypernym of the trigger (found in section 3.2).

We should point out that some of the same effects can be obtained by making patterns more general, i.e., adding "optional" attributes (i.e., optional sub-patterns), etc. Nonetheless, the pattern mutation is more efficient because it will automatically learn such generalization on an as-needed basis in an entirely data-driven fashion, while also maintaining high precision of the resulting pattern set. It is thus a more general method. Figure 4 illustrated the use of the elements combination technique. In this example,

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


EX: In 1991, he was arrested in Kissimmee on a charge of fondling a child under age 16.
Matches the text but misses Place role
Doesn't match text (Except for Place)
Import Place Sub-pattern
Pattern-1
Pattern-2
Derived new pattern

```
Trigger: arrest_V                                                          Pattern-1
Person: <N (obj, PER): Person> <V(arrest): trigger>
Crime:   <V(arrest): trigger> <Prep> <N><Prep><N><Prep(of)><V(Crime): Crime>
Time-Within:   <Prep(in)><N(timex2): Time-Within><SentHead><V(arrest): trigger>

Trigger: arrest_V                                                          Pattern-2
Person: <N (obj, PER): Person> <V(arrest): trigger>
Place:   <V(arrest): trigger> <Prep(in)> <N(GPE): Place>
Time-Within:   <Prep(on)><N(timex2): Time-Within><SentHead><V><V(arrest): trigger>

Trigger: arrest_V                                                Derived new pattern
Person: <N (obj, PER): Person> <V(arrest): trigger>
Place:   <V(arrest): trigger> <Prep(in)> <N(GPE): Place>
Crime:   <V(arrest): trigger> <Prep> <N><Prep><N><Prep(of)><V(Crime): Crime>
Time-Within:   <Prep(in)><N(timex2): Time-Within><SentHead><V(arrest): trigger>
```

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

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 a verification step explained in a later section.

3.5 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, by computing alternative trigger-attribute paths in the dependency tree. The new pattern (shown in Figure 5B) is of course subject to confidence validation, after which it will be immediately applied to extract more events.

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 a variant of an argument advanced in (Gale, et al. 2000) that a polysemous word used multiple times within a single document, is consistently used in the same sense. So if we extract an event mention (of type T) with trigger *t* in one part of a document, and then find that *t* occurs in another part of the same document, then we may assume that this second occurrence of *t* has the same sense as the first. Since *t* is a trigger for an event of type T, we can hypothesize its subsequent occurrences indicate additional mentions of type T events that were not extracted by any of the existing patterns. Our objective is to exploit these unextracted mentions and then automatically generate additional event patterns.

Indeed, Ji (2008) showed that trigger co-occurrence helps finding new mentions of the

Pattern ID: 1207
Type: Conflict **Subtype:** Attack
Trigger: bombing_N
Target: <N(bombing): trigger> <Prep(of)> <N(FAC): Target>
Attacker: <N(PER): Attacker> <V> <N(bombing): trigger>
Time-Within: <N(bombing): trigger> <Prep> <N> <Prep> <N> <E0> <V> <N(timex2): Time-within>

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

Pattern ID: 1286
Type: Conflict **Subtype:** Attack
Trigger: attack_N
Target: <N(FAC): Target> <Prep(in)> <N(attack): trigger>
Attacker: <N(PER): Attacker> <V> <N> <Prep> <N> <Prep(in)> <N(attack): trigger>
Time-Within: <N(attack): trigger> <E0> <V> <N(timex2): Time-within>

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

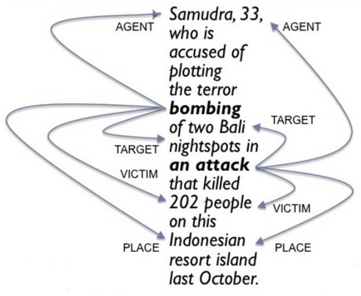

Samudra, 33, who is accused of plotting the terror bombing of two Bali nightspots in an attack that killed 202 people on this Indonesian resort island last October.
AGENT
AGENT
TARGET
TARGET
VICTIM
VICTIM
PLACE
PLACE

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

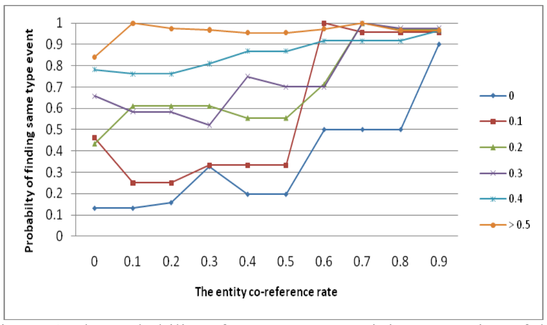


Figure 6. The probability of a sentence containing a mention of the same type of event within a single document

same event; however, we found that if using entity co-reference as another factor, more new mentions could be identified when the trigger has low projected accuracy (Liu, 2009; Yu Hong, et al. 2011). Our experiments (Figure 6[4]), which compared the triggers and the roles across all event mentions within each document on ACE training corpus, showed that when the trigger accuracy is 0.5 or higher, each of its occurrences within the document indicates an event mention of the same type with a very high probability (mostly > 0.9). For triggers with lower accuracy, this high probability is only achieved when the two mentions share at least 60% of their roles, in addition to having a common trigger. Thus our approach uses co-occurrence of both trigger and event argument for detecting new event mentions.

In Figure 7, an End-Position event is extracted from left sentence (L), with "***resign***" as the trigger and "***Capek***" and "***UBS***" assigned Person and Entity roles, respectively[5]. The right sentence (R), taken from the same document, contains the same trigger word, "***resigned***" and also the same

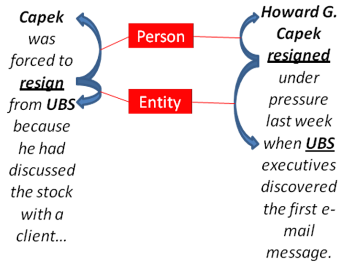


Figure 7. Two event mentions have different triggers and sub-patterns structures

Pattern ID: -1
Type: Personnel **Subtype:** End-Position
Trigger: resign_V
Person: <N(PER, subj): Person> <V(resign): trigger>
Entity: <V(resign):trigger> <E0> <N(ORG): Entity> <N> <V>

Figure 7A. A new pattern for *End-Position* learned by exploiting event co-reference.

[4] The X-axis is the percentage of entities coreferred between the EVMs (Event mentions) and the SEs (Sentences); while the Y-axis shows the probability that the SE contains a mention that is the same type as the EVM.

[5] Entity is the employer in the event

entities, "***Howard G. Capek***" and "***UBS***". The projected accuracy of *resign*_V as an End-Position trigger is 0.88. With 100% argument overlap rate, we estimate the probability that sentence R contains an event mention of the same type as sentence L (and in fact co-referential mention) at 97% (We set 80% as the threshold). Thus a new event mention is found and a new pattern for End-Position is automatically derived from R, as shown in Figure 7A.

3.6 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 based on the similarity between the extracted events and human annotated events:

- *A Full match* is achieved when the event type is correctly identified and all its roles are correctly matched. A full credit is added to the pattern score.
- *A Partial match* is achieved when the event type is correctly identified but only a subset of roles is correctly extracted. A partial score, which is the ratio of the matched roles to the whole roles, is added.
- *A False Alarm* occurs when a wrong type of event is extracted (including when no event is present in text). No credit is added to the pattern score.

In the subsequent steps, the validation is extended over parts of the unannotated corpus. In Riloff (1996) and Sudo et al. (2001), the pattern accuracy is mainly dependent on its occurrences in the relevant documents[6] vs. the whole corpus. However, one document may contain multiple types of events, thus we set a more restricted validation measure on new rules:

- *Good Match* If a new rule "rediscovers" already extracted events of the same type, then it will be counted as either a *Full Match* or *Partial Match* based on previous rules
- *Possible Match* If an already extracted event of same type of a rule contains same entities and trigger as the candidate extracted by the rule. This candidate is a possible match, so it will get a partial

[6] If a document contains same type of events extracted from previous steps, the document is a relevant document to the pattern.

Event id: 27
from: sample
Projected Accuracy: 0.1765
Adjusted Projected Accuracy: 0.91
Type: Justice **Subtype:** Arrest-Jail
Trigger: capture
Person sub-pattern: <N(obj, PER): Person> <V(capture): trigger>
Co-occurrence ratio: {para_Conflict_Demonstrate=100%, …}
Mutually exclusive ratio: {sent_Conflict_Attack=100%, para_Conflict_Attack=96.3%, …}

Figure 8. An *Arrest-Jail* pattern with context profile information

score based on the statistics result from Figure 6.

- *False Alarm* If a new rule picks up an already extracted event in different type

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

For the patterns where the projected accuracy score falls under the cutoff threshold, we may still be able to make some "repairs" by taking into account their context profile. To do so, we applied a similar approach as (Liao, 2010), which showed that some types of events can appeared frequently with each other. We collected all the matches produced by such a failed pattern and created a list of all other events that occur in their immediate vicinity: in the same sentence, as well as the sentences before and after it[7]. These other events, of different types and detected by different patterns, may be seen as co-occurring near the target event: these that co-occur near positive matches of our pattern will be added to the *positive context support* of this pattern; conversely, events co-occurring near false alarms will be added to the *negative context support* for this pattern. By collecting such contextual information, we can find contextually-based indicators and non-indicators for occurrence of event mentions. When these extra constraints are included in a previously failed pattern, its projected accuracy is expected to increase, in some cases above the threshold.

For example, the pattern in Figure 8 has an initially low projected accuracy score; however, we find that positive matches of this pattern show a very high (100% in fact) degree of correlation with mentions of *Demonstrate* events. Therefore, limiting the application of this pattern to situations where a *Justice-Arrest-Jail* event is mentioned in a nearby text improves its projected accuracy to 91%, which is well above the required threshold.

In addition to the confidence rate of each new pattern, we also calculate projected accuracy of each of the role sub-patterns, 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. To validate a sub-pattern we apply it to the training corpus and calculate its projected accuracy score by dividing the number of correctly matched roles by the total number of matches returned. The projected accuracy score will tell us how well a sub-pattern can distinguish a specific event role from other information, when used independently from other elements of the complete pattern.

Figure 9 shows three sub-pattern examples. The first sub-pattern extracts the *Victim* role in a *Life-Die* event with very high projected accuracy. This sub-pattern is also a good candidate for generations of additional patterns for this type of event, a process which we describe in section D. The second sub-pattern was built to extract the *Attacker* role in Conflict-Attack events, but it has very low projected accuracy. The third one shows another *Attacker* sub-pattern whose projected accuracy score is 0.417 after the first step

Victim pattern: <N(obj, PER): Victim> <V(kill): trigger> (*Life-Die*)
Projected Accuracy: 0.9390243902439024
Number of negative matches: 5
Number of Positive matches: 77

Attacker pattern: <N(subj, PE/PER/ORG): Attacker> <V> <V(use): trigger> (*Conflict-Attack*)
Projected Accuracy: 0.025210084033613446
Number of negative matches: 116
Number of positive matches: 3

Attacker pattern: <N(subj, GPE/PER): Attacker> <V(attack): trigger> (*Conflict-Attack*)
Projected Accuracy: 0.4166666666666667
Number of negative matches: 7
Number of positive matches: 5

categories of positive matches:	GPE: 4 GPE_Nation: 4 PER: 1 PER_Individual: 1
categories of negative matches:	GPE: 1 GPE_Nation: 1 PER: 6 PER_Group: 1 PER_Individual: 5

Figure 9. sub-patterns with projected accuracy scores

[7] If a known event is detected in the same sentence (sent_…), the same paragraph (para_…), or an adjacent paragraph (adj_para_...) as the candidate event, it becomes an element of the pattern context support.

Table 1. Sub-patterns whose projected accuracy is significantly increased after noisy samples are removed

Sub-patterns	Projected Accuracy	Additional constraints	Revised Accuracy
Movement-Transport:			
<N(obj, PER/VEH): Artifact> <V(send): trigger>	0.475	removing PER	0.667
<V(bring): trigger> <N(obj)> <Prep = to> <N(FAC/GPE): Destination>	0.375	removing GPE	1.0
…			
Conflict Attack:			
<N(PER/ORG/GPE):Attacker><N(attack):trigger>	0.682	removing PER	0.8
<N(subj,GPE/PER):Attacker><V(attack): trigger>	0.417	removing GPE	0.8
<N(obj,VEH/PER/FAC):Target><V(target):trigger>	0.364	removing PER_Individual	0.667
…			

in validation process. This is quite low; however, it can be repaired by constraining its entity type to GPE. This is because we note that with a GPE entity, the subpattern is 80% on target, while with PER entity it is 85% a false alarm. After this sub-pattern is restricted to GPE its projected accuracy becomes 0.8.

Table 1 lists example sub-patterns for which the projected accuracy increases significantly after adding more constrains. When the projected accuracy of a sub-pattern is improved, all patterns containing this sub-pattern will also improve their projected accuracy. If the adjusted projected accuracy rises above the predefined threshold, the repaired pattern will be saved.

In the following section, we will 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.

4 Evaluation

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 split ACE training corpus[8] randomly into 5 folders and trained BEAR on the four folders and evaluated it on the left one. Then, we did 5 fold cross validation. Our experiments showed that BEAR reached the best cross-validated score, 66.72%, when pattern accuracy threshold is set at 0.5. The highest score of single run is 67.62%. In the following of this section, we will use results of one single run to display the learning behavior of BEAR.

In Figure 10, X-axis shows values of the learning threshold (in descending order), while Y-axis is the average F-score achieved by the automatically learned patterns for all types of events against the test corpus. The red (lower) line represents BEAR's base run immediately after the first iteration (supervised learning step); the blue (upper) line represents BEAR's performance after an additional 10 unsupervised learning cycles[9] are completed. We note that the final performance of the bootstrapped system steadily increases as the learning threshold is lowered, peaking at about 0.5 threshold value, and then declines as the threshold value is further decreased, although it remains solidly above the base run. Analyzing more closely a few selected points on this chart we note, for example, that the base run at threshold of 0 has F-score of 34.5%, which represents 30.42% recall, 40% precision. On the other end of the curve, at the threshold of 0.9, the base run precision is 91.8% but recall at only 21.5%, which produces F-score of 34.8%. It is interesting to observe that at neither of these two extremes the system learning effectiveness is particularly good, and is significantly less than at

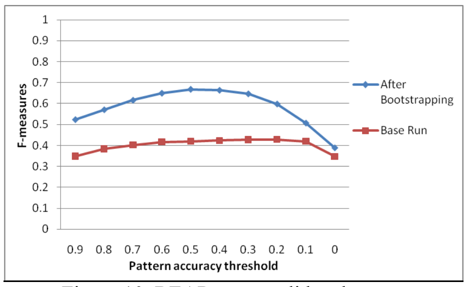


Figure 10. BEAR cross-validated scores

[8] ACE training data contains 599 documents from news, weblog, usenet, and conversational telephone speech. Total 33 types of events are defined in ACE corpus.

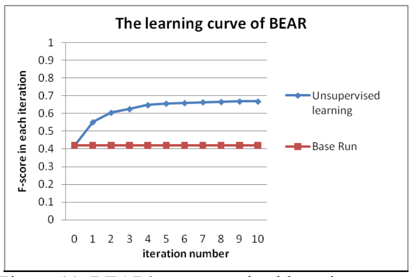


Figure 11. BEAR's unsupervised learning curve.

[9] The learning process for one type of event will stop when no new patterns can be generated, so the number of learning cycles for each event type is different. The highest number of learning cycles is 10 and lowest one is 2.

the median threshold of 0.5 (based on the experiments conducted thus far), where the system performance improves from 42% to 66.86% F-score, which represents 83.9% precision and 55.57% recall.

Figure 11 explains BEAR's learning effectiveness at what we determined empirically to be the optimal confidence threshold (0.5) for pattern acquisition. We note that the performance of the system steadily increases until it reaches a plateau after about 10 learning cycles.

Figure 12 and Figure 13 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.

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

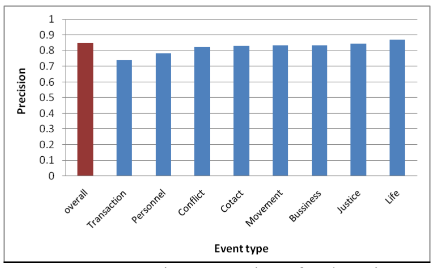


Figure 12. Event mention extraction after learning: precision for each type of event

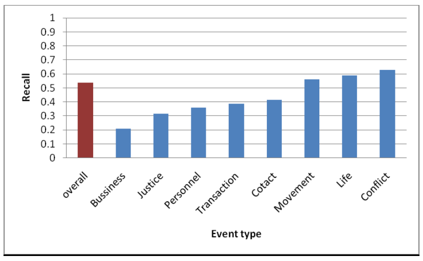


Figure 13. Event mention extraction after learning: recall for each type of event

Table 2. BEAR performance following different selections of learning steps

	Precision	Recall	F-score
Base1	0.89	0.22	0.35
Base2	0.87	0.28	0.42
All	0.84	0.56	0.67
PMM	0.84	0.48	0.61
CBM	0.86	0.37	0.52

see how they contribute to the end performance. Base1 and Base2 showed the result without and with adding trigger synonyms in event extraction. By introducing trigger synonyms, 27% more good events were extracted at the first iteration and thus, BEAR had more resources to use in the unsupervised learning steps.

The ALL is the combination of PMM and CBM, which demonstrate both methods have the contribution to the final results. Furthermore, 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.

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. Our evaluation using ACE dataset demonstrated that that bootstrapping can be effectively applied to learning event extraction rules for 33 different types of events and that the resulting system can outperform supervised system (base run) significantly.

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 a 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.

References

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

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

Hong, Y., Zhang, J., Ma, B., Yao, J., Zhou, G., and Zhu, Q,. 2011. Using Cross-Entity Inference to Improve Event Extraction. In *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL 2011).* Portland, Oregon, USA.

Ji, H. and Grishman, R. 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.

Liao, S. and Grishman R. 2010. Using Document Level Cross-Event Inference to Improve Event Extraction. *In Proc. ACL-2010,* pages 789-797, Uppsala, Sweden, July.

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

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

Riloff, E. 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.

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

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

Strzalkowski, T., and Wang, J. 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

Thelen, M., Riloff, E. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10.* 214-222. Morristown, NJ: Association for Computational Linguistics

Xu, F., Uszkoreit, H., &amp; Li, H. (2007). A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In Proc. of the 45th Annual Meeting of the Association of Comp. Linguistics, pp. 584–591, Prague, Czech Republic.

Yangarber, R., and Grishman, R. 1998. NYU: Description of the Proteus/PET System as Used for MUC-7 ST. In *Proceedings of the 7th conference on Message understanding.*

Yangarber, R., Grishman, R., Tapanainen, P., and Huttunen, S. 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

Yarowsky, D. 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