

# Transformation Based Learning for Specialization of Generic Event Extractions

Mary D. Taffet, Nancy J. McCracken, Eileen E. Allen, Elizabeth D. Liddy  
Syracuse University  
School of Information Studies  
Center for Natural Language Processing (CNLP)  
4-206 Center for Science & Technology  
Syracuse, NY 13244  
December 2002  
{mdtaffet@syr.edu, njm@ecs.syr.edu, eeallen@syr.edu, liddy@syr.edu}

## Abstract:

As part of our Evidence Extraction and Link Discovery (EELD) project, we proposed to use Transformation Based Learning (TBL) to learn domain-specific specializations for generic event extractions. The primary goal of our learning task was to reduce the amount of human effort required for specializing generic event extractions to domains that are new and specific. Three initial annotation cycles and one annotation review and correction cycle involving a total of 70 documents were completed, with slightly over 32 hours required for the entire annotation effort; where possible, the annotation cycles started with bootstrapped files resulting from the application of TBL rules learned after the prior annotation cycle. A five-fold evaluation was completed using the annotated files as the gold standard for evaluation purposes. When our analysis was limited to specialized event types with 10 or more examples available for training, we achieved 67.93% Coverage, 88.93% Accuracy, and an F-score of 77.02%. Several conclusions can be drawn from our study: (1) the use of TBL to learn specializations of generic extractions to specific domains is possible, (2) the use of TBL leads to a significant reduction in the human effort involved in specializing to a new domain, (3) sparsity of training data has a large impact on the results of learning, and (4) with more training instances, coverage and accuracy would improve.

## Motivation:

In the early days of Information Extraction (IE), IE systems extracted a limited amount of information in order to fill in the blanks in a predefined domain-specific template. Today's IE systems increasingly rely on a much broader model of generic event, entity and relation extraction in order to capture a wide variety of useful information in various domains. While today's generic extractions are abundant and useful, extractions that have been specialized for the domain are generally more meaningful and more easily interpreted by those who must make sense of the information provided by IE systems. The challenge to today's IE systems is to extract a wide variety of useful information about events, entities and relations while at the same time adapting those extractions to the domain at hand; this is the largest single impediment to portability of today's IE systems due to the effort required for this specialization.

As part of our Evidence Extraction and Link Discovery (EELD) project, we proposed to use Transformation Based Learning (TBL) to learn domain-specific specializations for generic event extractions. The primary goal of our learning task was to reduce the amount of human effort required for specializing generic event extractions to domains that are new and specific.

## eQuery and Generic Extraction:

The eQuery system created by the Center for Natural Language Processing (CNLP) is a Natural Language Processing-based automatic IE system which implements algorithms that interpret language at all the levels at

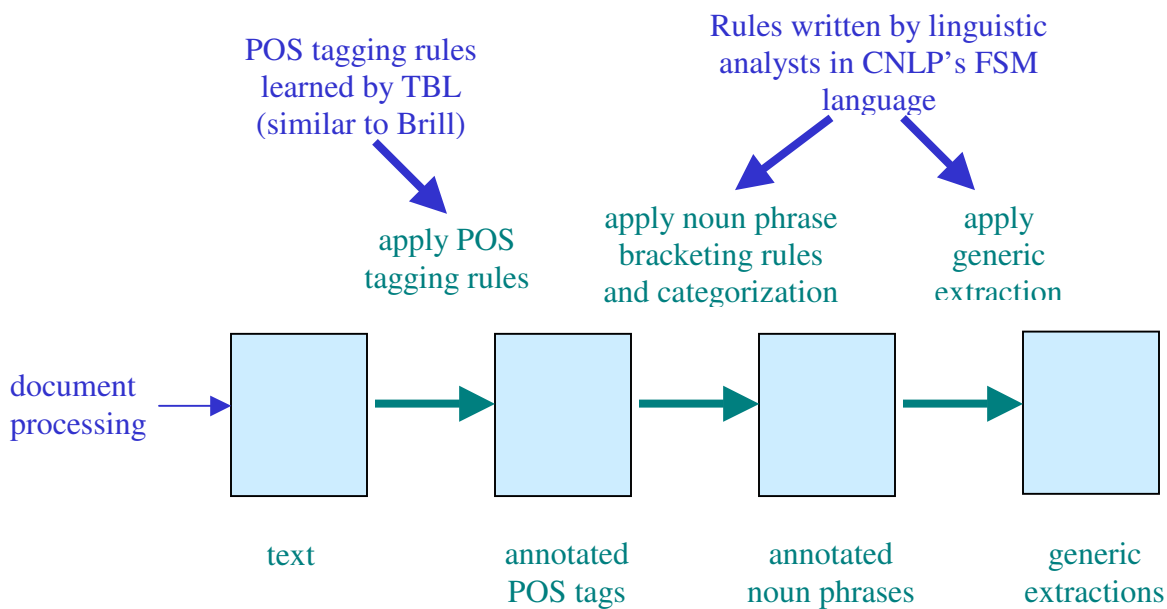
which humans are known to extract meaning. These are the morphological, lexical, syntactic, semantic, discourse and pragmatic levels. Using these levels eQuery extracts important concepts and relations from texts by applying sophisticated and proven natural language processing techniques.

CNLP extracts entities (which includes named entities) and events from text documents and represents them as objects in extraction tables. These entities and events are put into frames with modifying slots. One of the primary slots is the “type” slot, which gives the type of the entity or event. Some slots can be viewed as attributes, while others are viewed as relations, primarily if their value is another entity. The attributes and relations are dynamically assigned as appropriate to entities and events during the extraction process.

For named entities, attributes and relations are grouped by “episode”, corresponding approximately to the different mentions of the entity. So each relation will be in a separate episode and can have attributes, such as point-in-time, of that relation.

The extracted entities include proper noun phrases and compositional noun phrases, except those denoting numeric concepts, which are treated separately. The generically extracted events are almost all verbs, particularly those denoting actions, but not including those which denote states of being where the information is put into an attribute or relation instead.

Prior to our work on TBL for specialization, our system had the following structure:



**Fig. 1: Structure of eQuery prior to specialization**

The generic extraction process uses the marked-up text to identify events and entities based on surface lexical and syntactic clues. For each document, the collection of events and entities are saved in an extraction table. Of most interest to us for this study are the events and certain entities which are event-like nominalizations. For these events and event-like entities, our table elements use a case frame representation to store information about the various roles associated with the event; the case frames are based on the concept of case grammar introduced by Fillmore (Fillmore, 1968). The case frames capture the relationships between events and entities that exist at a semantic, conceptual level, regardless of the surface syntactic structure.

Information about the roles associated with each event are stored in slots which are attribute-value pairs consisting of a slot name and a slot value. For generic extraction, our slot names are generic in nature; they include about 20 generic role names from Sowa's conceptual graph theory (Sowa, 1984), as shown in Table 1.

**Table 1: Generic role names from Sowa:**

agent	duration	material	part-of
cause	frequency	measure	point-in-time
characteristic	instrument	method	quantity
content	location	negation	recipient
destination	manner	object	source

We also have at least 24 additional generic relation names as shown below in Table 2, and more continue to be added over time.

**Table 2: Additional generic relation names:**

acronym	charge	geographic-affiliation
affiliation	condition	isa
age	cost	linked
alias	date	participants
amount	date-of-birth	price
area	date-of-death	prior-to
associated	distance	purpose
body-part	following	reason

An example of extracted entities and events are shown below in Figure 2.

**Sentence:**

*Just today, a business near the French Embassy was blown up and four people were killed by an unidentified gunman.*

**Generic extractions:**

id = 35  
 frametype = namedentity  
 text = French Embassy  
 type = spfac  
 Episode 0  
   sentenceid = s28  
   mention = French Embassy

id = 128  
 frametype = entity  
 sentenceid = s28  
 text = gunman  
 Episode 0  
   characteristic = unidentified  
   sentenceid = s28

id = 129  
frametype = event  
sentenceid = s28  
text = kill  
Episode 0  
agent = unidentified gunman  
object = four people  
sentenceid = s28

id = 131  
frametype = event  
sentenceid = s28  
text = blow\_up  
Episode 0  
object = French Embassy = 35  
sentenceid = s28

id = 132  
frametype = entity  
sentenceid = s28  
text = people  
Episode 0  
amount = four  
sentenceid = s28

id = 133  
frametype = entity  
sentenceid = s28  
text = today  
Episode 0  
characteristic = just  
sentenceid = s28

**Fig. 2: Example of Extracted Events and Entities**

Once extracted into this format, these events and entities can be queried, stored in databases for use by link discovery modules, and fed into visualizers, greatly facilitating access to information.

#### The Learning Task:

The learning task originally defined for this project is specialization of these generic event extractions for a specific domain. Specialization of generic event extractions involves varying levels of complexity. At the simplest level of complexity, specialization involves adding an event type and mapping the slot name to a different value that carries more conceptual meaning for the specific domain involved, as illustrated in Figure 3, which shows extractions from the same sentence as Figure 2, but after specialization has taken place.

#### **Sentence:**

*Just today, a business near the French Embassy was blown up and four people were killed by an unidentified gunman.*

## Extractions after specialization:

id = 35  
frametype = namedentity  
text = French Embassy  
type = spfac  
Episode 0  
    sentenceid = s28  
    mention = French Embassy

id = 128  
frametype = entity  
sentenceid = s28  
text = gunman  
Episode 0  
    characteristic = unidentified  
    sentenceid = s28

id = 129  
frametype = event  
sentenceid = s28  
text = kill  
**type = kill**  
Episode 0  
    **perpetrator** = unidentified gunman  
    **victim** = four people  
    sentenceid = s28

id = 131  
frametype = event  
sentenceid = s28  
text = blow\_up  
**type = attempt to kill**  
Episode 0  
    **victim** = French Embassy = 35  
    sentenceid = s28

id = 132  
frametype = entity  
sentenceid = s28  
text = people  
Episode 0  
    amount = four  
    sentenceid = s28

id = 133  
frametype = entity  
sentenceid = s28  
text = today  
Episode 0  
    characteristic = just  
    sentenceid = s28

### Fig. 3: Example of specialization for new domain

However, even this simpler form of event specialization can be complicated by word sense ambiguity. For example, not all instances of the event “pay” should be specialized; if the generic object is “attention” (pay attention) or “respects” (pay respects), then the event should not be specialized as this sense of the word “pay” falls outside of the domain model that includes the payment event.

A greater level of complexity is seen when an event frame is restructured to more appropriately capture the conceptual meaning for a specific domain, as shown in Figure 4.

#### Sentence:

*But in November Rachuk committed suicide in Russia, and it was some time later that a certain Sadykov asked the bosses of Summit International for a meeting.*

#### Extractions before restructuring:

id = 34  
frametype = namedentity  
text = Russia  
type = cntry  
Episode 0  
    sentenceid = s17  
    mention = Russia

id = 43  
frametype = namedentity  
text = Summit International  
type = co  
Episode 0  
    sentenceid = s17  
    mention = Summit International

#### Extractions after restructuring:

id = 34  
frametype = namedentity  
text = Russia  
type = cntry  
Episode 0  
    sentenceid = s17  
    mention = Russia

id = 43  
frametype = namedentity  
text = Summit International  
type = co  
Episode 0  
    sentenceid = s17  
    mention = Summit International

id = 47  
frametype = namedentity  
text = Vladimir Rachuk  
type = per  
Episode 0  
  lastname = Rachuk  
  firstname = Vladimir  
  isa = Russian financier  
  mention = Vladimir Rachuk  
  sentenceid = s13  
Episode 1  
  point-in-time = November = 49  
  mention = Rachuk  
  sentenceid = s17

id = 49  
frametype = namedentity  
text = November  
type = unknown  
Episode 0  
  sentenceid = s17  
  mention = November

id = 50  
frametype = namedentity  
text = Sadykov  
type = unknown  
Episode 0  
  characteristic = certain  
  mention = Sadykov  
  sentenceid = s17

id = 143  
frametype = event  
sentenceid = s17  
text = **commit**  
Episode 0  
  agent = November Rachuk  
  location = Russia = 34  
  point-in-time = November = 49  
  **object = suicide**  
  sentenceid = s17

id = 47  
frametype = namedentity  
text = Vladimir Rachuk  
type = per  
Episode 0  
  lastname = Rachuk  
  firstname = Vladimir  
  isa = Russian financier  
  mention = Vladimir Rachuk  
  sentenceid = s13  
Episode 1  
  point-in-time = November = 49  
  mention = Rachuk  
  sentenceid = s17

id = 49  
frametype = namedentity  
text = November  
type = unknown  
Episode 0  
  sentenceid = s17  
  mention = November

id = 50  
frametype = namedentity  
text = Sadykov  
type = unknown  
Episode 0  
  characteristic = certain  
  mention = Sadykov  
  sentenceid = s17

id = 143  
frametype = event  
sentenceid = s17  
text = **commit suicide**  
**type = kill**  
Episode 0  
  agent = November Rachuk  
  location = Russia = 34  
  point-in-time = November = 49  
  **object = null**  
  sentenceid = s17

id = 145  
frametype = event  
sentenceid = s17  
text = ask  
Episode 0  
agent = Sadykov = 50  
object = boss of Summit International  
sentenceid = s17

id = 146  
frametype = entity  
sentenceid = s17  
text = suicide  
Episode 0  
location = Russia = 34  
sentenceid = s17

id = 147  
frametype = entity  
sentenceid = s17  
text = boss  
Episode 0  
associated = Summit International = 43  
sentenceid = s17

id = 145  
frametype = event  
sentenceid = s17  
text = ask  
Episode 0  
agent = Sadykov = 50  
object = boss of Summit International  
sentenceid = s17

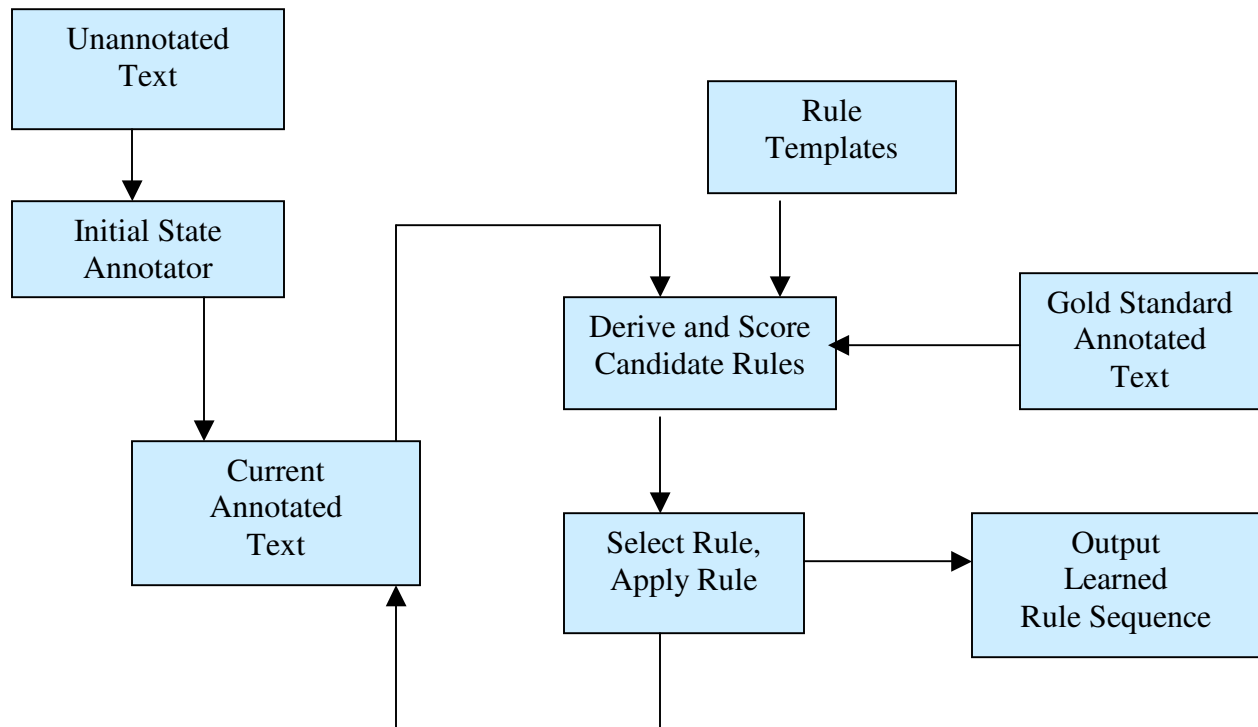
id = 146  
frametype = entity  
sentenceid = s17  
text = suicide  
type = kill  
Episode 0  
location = Russia = 34  
sentenceid = s17

id = 147  
frametype = entity  
sentenceid = s17  
text = boss  
Episode 0  
associated = Summit International = 43  
sentenceid = s17

**Fig. 4: Example of restructuring to fit new domain**

## What is Transformation Based Learning?

**Transformation-Based Error-Driven Learning (TBL)** is a robust corpus-based machine learning paradigm that is comprised of unannotated text, an initial state annotator that can be at any level of sophistication (Brill, 1993) but is typically based on a naïve and simplistic algorithm, a hand-tagged or hand-corrected annotated corpus considered to be the gold standard, a set of transformation templates, and an iterative learning program which learns the transformation rules necessary to change the annotations of the initial state annotator to match those found in the gold standard corpus. This approach is error-driven because the transformations learned at each step of the iteration are those that lead to the greatest reduction in errors when compared to the gold standard. The transformation-based error-driven learning paradigm is illustrated in Figure 5, adapted from (Ramshaw & Marcus, 1996b).



**Fig. 5: Transformation-Based Error-Driven Learning Paradigm**

## Why TBL vs. some other machine learning method?

Transformation-based, error-driven learning has the following advantages, TBL: (a) creates a relatively small number of rules that are linguistically motivated and understandable to both humans and machines, (b) exploits a wide range of symbolic vs. statistical linguistic regularities, (c) iteratively transforms an initial automatic imperfect annotation into one with fewer errors, (d) requires an order of magnitude fewer decisions than estimating the parameters of statistical models, (e) is surprisingly resistant to overtraining (Ramshaw & Marcus, 1996a), and (f) is more powerful than decision trees.

Transformation-based error-driven learning has been successfully applied to numerous NLP tasks, including learning rules for part-of-speech tagging (Brill, 1993; Brill, 1994; Brill, 1995); prepositional phrase attachment (Brill & Resnik, 1994; Yeh & Vilain, 1998); subordinate conjunction attachment (Yeh & Vilain, 1998); parsing (Brill, 1993; Satta & Brill, 1996); word segmentation (Palmer, 1997; Hockenmaier & Brew, 1998); and grammatical relation extraction (Ferro, Vilain, & Yeh, 1999).

## The TBL Learning Environment for the EELD Project

In order to use TBL, several things must be prespecified: the initial state annotation algorithm, the search space, the list of allowable transformation templates and the scoring function. The initial state annotation algorithm specifies what annotations will be added by the initial state annotator. The search space defines both what elements can be referred to in transformations (e.g., words, word classes, part of speech tags, phrase types, etc.) and some measure of the maximum distance to be considered (e.g., 1 element away, 2 elements away, 3 elements away, etc.). The allowable transformation templates include the original content of the base element to be changed, the revised content after the change, and a description of the conditioning environment consisting of other elements and their distance from the base element, all in a sufficiently generic language that the template is considered to be uninstantiated (i.e. based on variable names). The scoring function defines the measures that will be used to determine the value of the net gain for a particular transformation; it is the comparison of these net gain values upon which the selection of the next transformation to be learned depends. The scoring function metrics are also used to determine when learning should stop; learning typically stops for a particular training run when either (1) there is no possible transformation left which leads to an additional net gain (reduction in errors), or (2) the remaining possible transformations do not lead to a net gain greater than the defined threshold.

The fnTBL toolkit was used for this learning task. Generic extraction tagged text provides the initial state annotation. The search space and allowable transformation templates are both user-defined aspects of the fnTBL toolkit, and are defined in the rule template file. One example template from our rule template file is as follows:

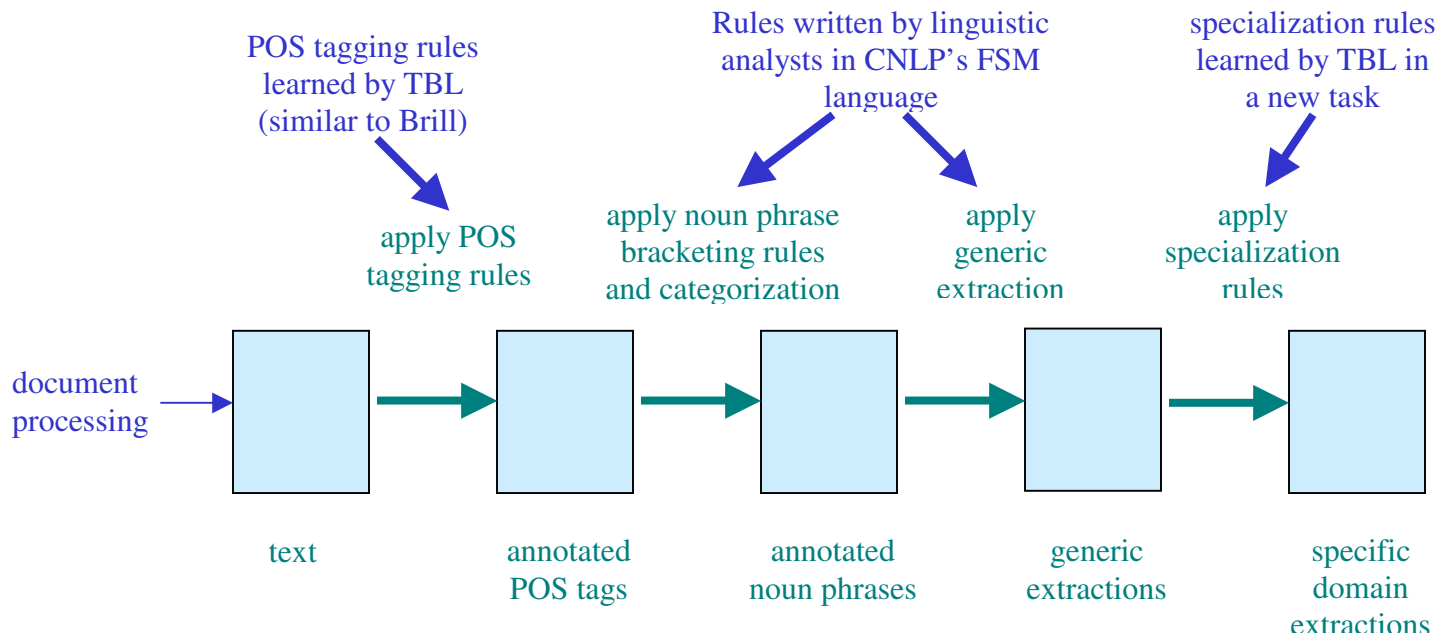
```
FrameType TextWord SlotWord SlotCat SlotName => SlotName
```

This template is used to learn the specialization rules for generic slot names. The scoring function is based on the number of good rule applications and the number of bad rule applications. The score of a rule learned with the fnTBL toolkit is the residual value of good – bad rule applications, and represents a net gain in learning. When running a learning cycle using the fnTBL toolkit, a threshold value is supplied by the developer. For the fnTBL toolkit, the threshold value indicates a learning cutoff such that the net gain in learning must be one more than the value of the threshold.

## Detail on use of TBL for EELD

### *Addition of Transform phase*

For this project, an additional processing module was added to our eQuery document processing system. This additional module applies specialization rules to the generic event extractions, as seen in the far right of the figure. The specialization rules, also called transform rules, are the result of the TBL learning process.



**Fig. 6: Structure of eQuery after addition of specialization phase**

Specialization, or transform rules, include two components – a triggering environment and a rewrite rule:

- if [triggering environment] then [re-write rule]
- if [event = *kill*, *object* = ?X, ?X.type = person] then [object-> *victim*]

### *Training and Testing*

The data used for this study came from the Russian Contract Killing corpus supplied for the EELD project. Our initial effort was focused on obtaining a set of gold standard documents to use for training and testing purposes.

To this end, several annotation cycles were run. The first annotation cycle was purposefully small, and based on 14 sentences that were hand-picked by the analysts as representative of the domain-specific events that they wished to specialize; these sentences contained examples of such domain-specific events as *kill*, *murder* and *apprehend* among others. The analysts provided the generic extractions from these sentences along with the specialized extractions for these sentences. Using the generic extractions as the baseline and the matching specialized extractions as the gold standard; from these first 14 sentences, 26 specialization rules were learned using the fnTBL toolkit and a learning threshold of 0. A value of zero for the threshold was used to bootstrap our gold standard annotations.

For the second annotation cycle, 35 documents were chosen from the available corpus using a frequency count of the desired events. The documents in the corpus were examined automatically to determine the number of instances in each document of the following events:

**Table 3: Event Types and Associated Events for 2<sup>nd</sup> Annotation Cycle**

<b>Event Type</b>	<b>Associated Events</b>
arrest	arrest
detain	apprehend, detain
kidnap	abduct, kidnap
kill	assassinate, execute, kill, murder

A total event count for each document was obtained and the documents were sorted in decreasing order of frequency of this total event count. Without modification, this method resulted in a preference for the largest documents in the corpus; they corresponded to the largest values for total event count. Therefore, the method was modified to normalize the event frequency count for the size of the document. A list of documents was produced, showing document names in descending order of total event count; the first 35 documents on the list were chosen to be the first annotation corpus.

The initial set of 26 transform rules were applied to this set of 35 documents to bootstrap the annotation for the gold standard. Then the documents, along with both the generic extractions and the bootstrapped transformed extractions, were provided to the analysts for hand-correction of the preliminary transformed extractions. During this hand-correction effort, the analysts added event types where they were needed, specialized slot names where applicable, restructured event frames where necessary, and corrected the generic event extraction errors. The hand-correction effort was completed in 8 hours. The generic extractions and matching gold standard extractions from this set of 35 documents were fed into the fnTBL toolkit; again using threshold 0, and 272 transform rules were produced.

At that point, it was determined that more data was needed for training, so a third annotation cycle was included. For the third annotation cycle, another 35 documents were chosen using the method for the second annotation cycle, and the following set of events of interest:

**Table 4: Event Types and Associated Events for 3<sup>rd</sup> Annotation Cycle**

<b>Event Type</b>	<b>Associated Events</b>
arrest	arrest, charge
attempt-to-kill	attempt to assassinate, attempt to kill, attempt to murder
deal	bargain
detain	apprehend, detain, extradite
disappear	disappear, escape
investigation	interrogate, investigate, probe
kidnap	abduct, kidnap
kill	assassinate, execute, kill, murder
payment	invest, pay
telephone conversation	answer phone
theft	steal, theft
warn	alert, warn

As in the second annotation cycle, the rules learned from the prior cycle (in this case 272 rules) were applied to this set of 35 documents, once again in an effort to bootstrap the annotation of the gold standard. The resulting set of documents, generic extractions and preliminary specialized extractions was turned over to the analysts for hand-correction, and again, the hand correction was completed in 8 hours.

At this point, we had a combined set of 70 gold standard documents; the annotation cycles were complete and we began to focus on the learning cycles.

We ran a five-fold cross-validation test using an 80/20 split over the whole set of 70 documents. For each test, 56 documents were used for training and 14 documents were held out for testing.

To begin this process, each of the two sets of 35 documents was divided into 5 chunks of 7 documents each, using the alphabetical order of document names as the grouping factor; since the alphabetical order of document names had no bearing on the frequency of event counts, it was felt that alphabetical order allowed for sufficient randomness. To create the training and testing breakdown for each test run, one chunk from each set of 35 documents (two chunks for a total of 14 documents) was held out for testing, and the remaining four chunks from each set (eight chunks for a total of 56 documents) were used for training. For each test, a different group of chunks was held back for testing purposes.

For each cross-validation test, the method was the same:

- Run generic extraction only on the test set and save the output (i.e. extractions) to establish the baseline files
- Use the training set to run 3 different TBL learning cycles with the fnTBL toolkit
  - Using threshold 1
  - Using threshold 2
  - Using threshold 3
- Convert the rules learned by the fnTBL toolkit to eQuery format
- Apply the learned rules (transforms) to the test set and save the output (extractions) to facilitate evaluation
- Evaluate the learned rules by comparing for each test set the extractions from the baseline files, the transformed files, and the matching gold standard files for each threshold.

After running just one of the cross-validation tests, it became clear that there were some inconsistencies in the gold standard annotation between the first set of 35 documents and the second set of 35 documents. So the entire set of 70 documents with gold standard annotations was reviewed by the analysts again in an effort to remove the inconsistencies that had been present. This review and reconciliation effort took 16 hours of analyst time. When the review of the 70 documents was complete, and the gold standard annotations had been reconciled, that one test was re-run and the remaining tests were also run.

### *Evaluation and Results*

In preparation for evaluation of the results, comparison files were created that showed, for each element of each frame to be evaluated, the corresponding values from the baseline file, the gold standard file and the transformed file.

The results were evaluated using two complementary but slightly different methods. The first method used was to calculate desired changes, learned changes and correctly learned changes. Desired changes were determined by comparing the baseline extraction to the gold standard extraction; if these values differed, it was considered to be a desired change. Learned changes were determined by comparing the baseline extraction to the transformed extraction; if these values differed, it was considered to be a learned change. Correctly learned changes were determined by comparing the transformed extraction to both the baseline and gold standard extractions; if the transformed extraction was different from the baseline extraction and the same as the gold

standard extraction, it was considered to be a correctly learned change. These three values were then used to calculate coverage and accuracy figures, which were in turn used to calculate an F-score:

$$\text{Coverage (C)} = \text{Correctly Learned Changes} / \text{Desired Changes}$$

$$\text{Accuracy (A)} = \text{Correctly Learned Changes} / \text{Learned Changes}$$

$$F = (2 * C * A) / (C + A)$$

The results obtained for the five cross-validation tests are shown below in Table 5:

**Table 5: Five-fold cross-validation test results**

Test	Threshold 1	Threshold 2	Threshold 3
Test 1	Coverage = <b>68.29%</b> Accuracy = 88.42% F-score = <b>77.06%</b>	Coverage = 64.50% Accuracy = <b>90.15%</b> F-score = 75.20%	Coverage = 64.50% Accuracy = <b>90.15%</b> F-score = 75.20%
Test 2	Coverage = <b>54.05%</b> Accuracy = 88.21% F-score = <b>67.03%</b>	Coverage = 50.33% Accuracy = 90.55% F-score = 64.70%	Coverage = 48.36% Accuracy = <b>92.08%</b> F-score = 63.41%
Test 3	Coverage = <b>64.71%</b> Accuracy = 87.38% F-score = <b>74.36%</b>	Coverage = 59.17% Accuracy = 86.80% F-score = 70.37%	Coverage = 56.40% Accuracy = <b>88.59%</b> F-score = 68.92%
Test 4	Coverage = <b>54.47%</b> Accuracy = <b>91.16%</b> F-score = <b>68.19%</b>	Coverage = 48.98% Accuracy = 90.60% F-score = 63.58%	Coverage = 45.73% Accuracy = 91.09% F-score = 60.89%
Test 5	Coverage = <b>66.12%</b> Accuracy = 83.16% F-score = <b>73.67%</b>	Coverage = 60.93% Accuracy = 85.77% F-score = 71.25%	Coverage = 59.56% Accuracy = <b>87.55%</b> F-score = 70.89%

As the threshold level increases, the coverage decreases and the accuracy increases; these findings are consistent with what others have reported. These figures for coverage, accuracy and F-score are comparable to what others have reported for similar tasks (Gildea & Palmer, 2002; Gildea & Hockenmaier, 2003).

The results reported above are based on an evaluation method that is similar to recall and precision in that the judgments are binary in nature; the judgments made for this first evaluation are also binary in nature – a change is either desired or not, a change is either learned or not.

But in fact, this learning task is not binary; it is instead ternary in nature. Using the same comparison files that show the baseline, gold standard and transformed value for each element in each frame to be evaluated, another set of evaluation figures was devised that shows a breakdown of the original set of figures. This second set of figures is based on the following viewpoint:

- Desired learning: For each element, the following (binary) possibilities exist
  - Learning is desired (gold standard not = baseline)
  - Learning is **NOT** desired (gold standard = baseline)
- Actual learning: For each element, the following (ternary) possibilities exist
  - No learning achieved (transformed = baseline)
  - Learning achieved (transformed not = baseline) and

- Learning is **CORRECT** (transformed = gold standard)
- Learning achieved (transformed not = baseline) but
  - Learning is **INCORRECT** (transformed not = gold standard)

Combining these two views, five<sup>1</sup> possible combinations exist:

**Table 6: Comparison of Desired Learning to Actual Learning**

Desired Learning	Actual Learning	Comparison
No learning desired	(A) No learning achieved	gold standard = baseline, transformed = baseline
	(B) Incorrect (spurious) learning	gold standard = baseline, transformed not = baseline transformed not = gold standard
Learning desired	(C) No learning achieved (misses)	gold standard not = baseline, transformed = baseline
	(D) Correct learning achieved	gold standard not = baseline, transformed not = baseline transformed = gold standard
	(E) Incorrect learning (errors)	gold standard not = baseline, transformed not = baseline transformed not = gold standard

Using this new breakdown of the data, the calculation of coverage and accuracy is redefined as follows:

$$\begin{aligned} \text{Coverage} &= D / (C + D + E) \\ &= \text{Correct} / (\text{Misses} + \text{Correct} + \text{Errors}) \end{aligned}$$

$$\begin{aligned} \text{Accuracy} &= D / (B + D + E) \\ &= \text{Correct} / (\text{Spurious} + \text{Correct} + \text{Errors}) \end{aligned}$$

Recalculation of our results using this new breakdown points out that the biggest cause of failure is misses, or cases where we did not learn anything but were supposed to. Three illustrative results are shown below in detail:

**Table 7: Results using new breakdown**

Combination	Test 1 Threshold 1	Test 1 Threshold 2	Test 2 Threshold 1
<b>A</b> → No learning desired, No learning achieved	9706	9710	10668
<b>B</b> → No learning desired, Incorrect (spurious) learning	11	7	15
<b>C</b> → Learning desired, No learning achieved (misses)	<b>95</b>	<b>112</b>	<b>192</b>
<b>D</b> → Learning desired, Correct learning	252	238	247
<b>E</b> → Learning desired, Incorrect learning (errors)	22	19	18
<b>Desired changes based on only one training instance</b>	<b>43</b>	<b>43</b>	<b>81</b>
<b>Desired changes based on no more than two training instances</b>		<b>43 + 8 = 51</b>	

As the threshold increases for Test 1, the cases of incorrect learning (B and E) decrease, but the misses (C) increase. The direct result of the increase in the threshold value is failure to learn rules which apply to a small

<sup>1</sup> It is not possible to learn correctly when no learning is desired, so that combination does not exist.

number of instances (i.e. for the fnTBL toolkit, a number of instances equal to or less than the threshold value). This explains both the decrease in B and E (incorrect learning) and the increase in C (misses).

This analysis points out that sparsity of usable training instances has an impact on the value of C (misses). Fully half of the 192 misses for Test 2, threshold 1 are explained by the fact that 81 desired changes for this test had only 1 training instance available in the training data; these 81 desired changes could not possibly be learned given threshold 1. Sparsity of the training data also explains nearly half of the misses for Test 1, threshold 1 (43 out of 95), and Test 1, threshold 2 (51 out of 112).

Looking back at our original figures for Coverage and Accuracy in Table 5, we find that our results improve if we limit our analysis to only the specialized event types with 10 or more examples available for training. Limiting our analysis to only these events gives us 67.93% Coverage, 88.93% Accuracy, and an F-score of 77.02%.

## Conclusions

Several conclusions can be drawn from our study:

- The use of TBL to learn specializations of generic extractions to specific domains is possible.
- The use of TBL leads to a significant reduction in the human effort involved in specializing to a new domain.
- Sparsity of training data has a large impact on the results of learning.
- With more training instances, coverage and accuracy would improve.

## References

1. Brill, E. (1993). A corpus-based approach to language learning (Ph.D. Thesis). Philadelphia, PA: Department of Computer and Information Science, University of Pennsylvania. Available at: <http://www.cs.jhu.edu/~brill/dissertation.ps>
2. Brill, E. (1994). Some advances in transformation-based part of speech tagging. Twelfth National Conference on Artificial Intelligence (AAAI-94) . Available at: [http://www.cs.jhu.edu/~brill/TAGGING\\_ADVANCES.ps](http://www.cs.jhu.edu/~brill/TAGGING_ADVANCES.ps)
3. Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. Computational Linguistics. Available at: <http://www.cs.jhu.edu/~brill/CompLing95.ps>
4. Brill, E., & Resnik, P. (1994). A rule-based approach to prepositional phrase attachment disambiguation. COLING 1994 . Available at: <http://www.cs.jhu.edu/~brill/pp-attachment.ps>
5. Ferro, L., Vilain, M., & Yeh, A. (1999). Learning transformation rules to find grammatical relations. Computational Natural Language Learning: A workshop at the 9th Conf. of the European Chapter of the Association for Computational Linguistics .

6. Fillmore, C. J. (1968). The case for case. Universals in Linguistic Theory (pp. 1-88). New York: Holt, Rinehart and Winston, Inc.
7. Gildea, D., & Hockenmaier, J. (2003). Identifying semantic roles using combinatory categorial grammar. 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP 2003) Association for Computational Linguistics.  
Available at: <http://www.cis.upenn.edu/~dgildea/gildea-emnlp03.pdf>
8. Gildea, D., & Palmer, M. (2002). The necessity of parsing for predicate argument recognition. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02) (pp. 239-246). Association for Computational Linguistics.  
Available at: <http://www.aclweb.org/anthology/P02-1031.pdf>
9. Hockenmaier, J., & Brew, C. (1998). Error-driven learning of Chinese word segmentation. 12th Pacific Conference of Language and Information (pp. 218-229). Singapore: Chinese and Oriental Languages Processing Society.  
Available at: <http://www.ltg.ed.ac.uk/~chrisbr/papers/hockenmaier-colips98.1/>
10. Palmer, D. P. (1997). A trainable rule-based algorithm for word segmentation. Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL97) Association for Computational Linguistics.  
Available at: <http://ssli.ee.washington.edu/ssli/people/palmer/papers/ac197.ps>
11. Ramshaw, L. A., & Marcus, M. P. (1996a). Exploring the nature of transformation-based learning. In J. L. Klavens, & P. Resnik (Eds.), The balancing act: Combining symbolic and statistical approaches to language (pp. 135-156). Cambridge, MA: MIT Press.
12. Ramshaw, L. A., & Marcus, M. P. (1996b). Exploring the nature of transformation-based learning. In J. L. Klavens, & P. Resnik (Eds.), The balancing act: Combining symbolic and statistical approaches to language (pp. 135-156). Cambridge, MA: MIT Press.
13. Satta, G., & Brill, E. (1996). Efficient transformation-based parsing. ACL 1996 Association for Computational Linguistics.  
Available at: [http://www.cs.jhu.edu/~brill/Eff\\_Pars.ps](http://www.cs.jhu.edu/~brill/Eff_Pars.ps)
14. Sowa, J. F. (1984). Conceptual Structures: Information processing in mind and machine (The Systems Programming Series). Reading, Massachusetts: Addison-Wesley Publishing Company.
15. Yeh, A. S., & Vilain, M. B. (1998). Some properties of preposition and subordinate conjunction attachments. 17th International Conference on Computational Linguistics and 36th Annual

Meeting of the Association for Computational Linguistics (COLING-ACL '98) Montreal, Canada:  
Association for Computational Linguistics.  
Available at: <http://xxx.lanl.gov/ps/cmp-lg/9808007>