Identifying Events using Similarity and Context

Dominic R. Jones and Cynthia A. Thompson
School of Computing
University of Utah
Salt Lake City, UT 84103
dominicj@cs.utah.edu, cindi@cs.utah.edu

Abstract

As part of our work on automatically building knowledge structures from text, we apply machine learning to determine which clauses from multiple narratives describing similar situations should be grouped together as descriptions of the same type of occurrence. Our approach to the problem uses textual similarity and context from other clauses. Besides training data, our system uses only a partial parser as outside knowledge. We present results evaluating the cohesiveness of the aggregated clauses and a brief overview of how this work fits into our overall system.

1 Introduction

Early work in natural language processing included ambitious research on the representation and use of information about commonly experienced situations (Schank and Riesbeck, 1981). The concept of a *script* was introduced in this research, to explain how people understand these situations and make inferences about them. A script is a stereotypical sequence of events that occur as part of a larger situation and can be used to infer missing details from a partial description of the larger occurrence, in essence providing a means for extracting information that is not actually present in a text.

Research on scripts includes demonstrations of hand built scripts (Cullingford, 1978) and sketchy scripts (DeJong, 1982) and the adjustment of hand-built scripts using a genetic algorithm (Mauldin, 1989). Work on learning *schemata* under constrained circumstances (Mooney and DeJong, 1985) pursues similar goals.

Our research has indicated that scripts may not explicitly occur in common types of text, such as newspaper stories or incident reports. Other research also appears to support this conclusion (Clark and Porter, 1995). Therefore, we are investigating event *correlations* as a more appropriate and extractable knowledge structure. In general, it appears that long event sequences do not reliably recur in our data. We instead look for reliable correlations between a small number of events.

Our goal is to automatically extract correlated events from text, using only a partial parser as outside information. To support this goal, we need to group clauses from distinct texts into coherent events, handling several sources of variety in descriptions of the same type of occurrence. Synonymy and abbreviations are two common contributors. A more important phenomenon is the existence of semantic categories keyed to the events themselves. A number of different objects may participate in an event, and yet have dissimilarities that place them in different conventional semantic categories. For example, a tree and a parked vehicle may both be collided with in different aircraft crashes, yet it is difficult to conceive of a reasonably specific semantic category that contains both. Each is a physical object, yet there are a large number of other physical objects that would not reasonably participate in a crash in the same way (books, hamburgers, and moons are a few examples).

As a result of these phenomena, conventional semantic lexicons, whether hand built or automatically generated, differ from our work in two regards. First, they group words, not clauses. Second, they use pre-defined semantic categories instead of contextual relevance.

Our answer to this problem is a technique that uses textual similarity and context from neighboring events to decide when to group clauses. The only outside resource we use is a partial parser. Our technique takes parsed text and partially built event sequences and uses them to group clauses that represent the same event.

In the remainder of this paper, we present a brief overview of the sequence learning system before describing how we create events. We also evaluate our event formation technique using human judges' ratings of the cohesiveness of the resulting events. Finally, we discuss areas of related work before concluding the paper.

2 An Event Correlation Learning System

To understand the event formation process, some background on our larger system is useful. That system, SPANIEL (*S*equence *P*rocessing and *AN*alysis for *I*nference *E*nhancement and *L*earning), attempts to learn ordered correlations between events from a corpus of parsed narratives.[1] We assume that the narratives all describe related, but not necessarily identical, situations. An event is a particular atomic occurrence described by a single clause. SPANIEL uses a statistical model to identify correlations. Our overall goal is to capture significant correlations between events. We provide an overview here, and will discuss the details in a future paper.

Briefly, SPANIEL uses a modified Markov model to capture correlations. The model attempts to capture a sequence identifying the significant events in a text. Our modification to Markov chaining captures the fact that events can be conditioned on a prior event that occurs several clauses distant, instead of on direct sequential dependence between events.

After the training data is parsed, SPANIEL creates a graph from individual clauses. This graph is meant to capture the semantically important information from each clause in the training texts. The nodes in the graph are created from the actual clauses, with the arcs in the graph indicating which clauses occur together in a text, and in what order. Events are built from the nodes in the graph, with the arcs providing the starting point for correlations between events.

We convert the parsed text to graph nodes using a frame representation. Each frame contains four slots. Three *primary* slots represent the agent, the action, and the theme of the clause. The agent and action slot fillers are straightforward: the agent slot takes a noun phrase that was tagged as the clause's subject by the parser, and the action slot takes the clause's verb phrase. The theme slot is more flexible, and can be filled by a noun phrase, prepositional phrase, or adjective phrase that follows the verb. Noun phrases tagged as the direct object have priority, but the first prepositional or adjective phrase can serve as the filler in the absence of a direct object.[2] A fourth slot is only filled if the system identifies a dependent clause through the presence of a clause marker or the absence of a theme (e.g. "She reported she was having control problems."). Words are not stemmed, except in two special cases described later.

The three primary slots are filled by a head and a modifier. The head is the head word of the appropriate phrase. Prepositional phrases also concatenate the preposition to the head. Verb phrases include all verbs from the phrase, along with particles. The modifier is a concatenation of all adjectives, adverbs, and other nouns. Two examples of sentences and the resulting frames are given in Figure 1.

We define three functions over the graph. These functions allow us to both detect correlations between events, and identify when nodes might represent the same event. First, the *cooccurrence* of two nodes, $C(N_i, N_j)$, is the number of narratives in which N_i occurs before N_j, and thus quantifies each arc. N_i and N_j do not have to be neighbors: any number of other clauses could have separated them. Cooccurrence is directional: $C(N_i, N_j)$ is not the same as $C(N_j, N_i)$. Next, two nodes N_i and N_j *match*, denoted $M(N_i, N_j)$, if their frames are identical.[3] Finally, two nodes N_i and N_j are *similar*, denoted $S(N_i, N_j)$, if they demonstrate a certain degree of textual similarity. This function plays an important role in event creation, and we define it in detail in the next section.

Given this graph, SPANIEL uses a beam search to detect correlations of multiple events by incrementally expanding a sequence of events that occur frequently together. As of this writing, most resulting correlations include only a pair of events, with rare sequences of three events. Our goal is to find such correlations with minimal domain-specific knowledge. Thus, SPANIEL must create events from individual nodes as it expands event sequences. We describe this *event formation* process, the heart of this paper, in the next section.

3 Event Construction and Revision

The goal of event formation is to identify nodes that can play the same role within a domain. Event formation starts with a single node, identified as a potential part of a correlation, and attempts to expand this simple event by adding additional nodes. This process is unsupervised, so we restrict the search by defining two criteria that nodes must meet to be included in the same event. First, the Similarity criterion states that each node in an event must be textually *similar* to at least one other node in the event. Therefore, a node must be similar to at least one node that is already part of an event in order to be added to the event. Second, the InSeq criterion states that each node in the event must occur, in the appropriate order, with at least one node in each of the event's immediate neighbors in the sequence.

Similarity is determined by a function $S(N_i, N_j)$ that indicates whether two nodes share sufficient textual fea-

[1]Following previous authors, we use *narrative* instead of *story* to refer to a text, indicating that we do not expect to find a full plot structure.

[2]Passive verbs result in rearrangement of the fillers.

[3]Identical frames do not guarantee identical text, as the frames are simplified versions of the original clauses.

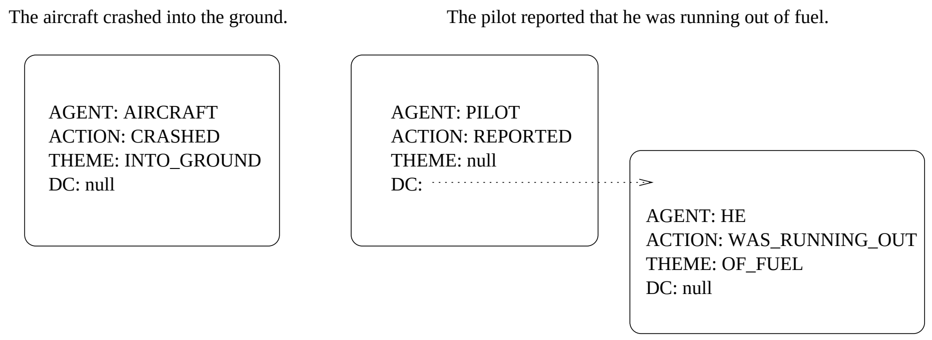

Figure 1: Example Frames

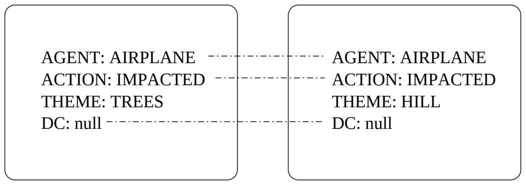

Figure 2: Textually Similar Nodes

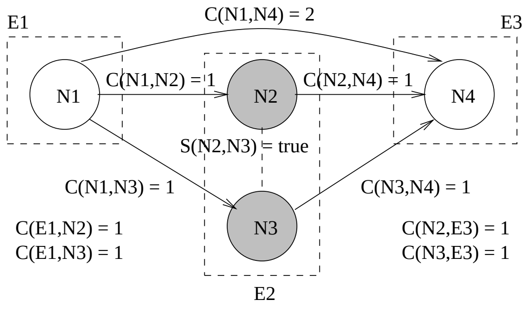

Figure 3: Example of Acceptable Node Pair

tures. Two nodes satisfy this function if two of their three primary slots share heads, with some exceptions described below. Empty fillers are not considered similar. In addition, the nodes must have similar dependent clauses (identified by recursively applying the same criteria). Null fillers for this slot are considered similar. An example of two textually similar nodes is given in Figure 2.

Two exceptions to this general definition handle nominalizations and "to be." First, if two nodes' actions do not match, but the stemmed head of one node's action matches the stemmed head of the other node's theme, the nodes are considered similar. Second, if the stemmed form of a node's action is "to be", the action is ignored in determining similarity. Therefore, the node's agent and theme must match the other node for the nodes to be similar.

The second criteria for adding nodes to events, InSeq, ensures that all nodes in an event occur in similar contexts. Within a correlation, we call events immediately prior to or following a given event its neighbors. Neighbors provide contextual information indicating which nodes occur in similar places in narratives. The InSeq constraint states that for a node to be added to an event, it must cooccur at least once with each of that event's neighbors in the proper order. To test for this cooccurrence, we extend the definition of cooccurrence to apply to mixed node and event arguments. Thus $C(N, E)$ or $C(E, N)$ is the number of times node N cooccurred in the training data with any node from event E. The order in which the arguments are given is the order in which they must appear to count.

An example of two nodes (N_2 and N_3) that meet our two criteria is given in Figure 3. The two nodes are both textually similar ($S(N_2, N_3) = true$) and have the same context ($C(E_1, N_2)$, $C(E_1, N_3)$, $C(N_2, E_3)$, and $C(N_3, E_3)$ are all greater than zero), so they can be combined into one event E_2.

Event formation is divided into two algorithms. The first, AUGMENT, adds nodes to an event based on the context provided by one neighbor. The second, REVISE, removes nodes that do not meet the InSeq criterion when an event acquires a second neighbor through further correlation expansion.

3.1 The AUGMENT Algorithm

The job of the AUGMENT algorithm is to form a new event and add it to the front or back of a correlation. Therefore the new event initially has one neighboring event. Each new event is initialized from a single node, and AUGMENT adds all nodes that meet both the Similarity and InSeq criteria to this node.

A phenomenon we refer to as *self-cooccurrence* complicates the augmentation process. This is when two nodes in the same event both occur in the same narrative. Since assigning two nodes to an event proposes that the nodes represent the same occurrence, self-cooccurrence is undesirable: it represents examples where the two

```
Input: $seed$, a seed node, $B$, a neighbor event
Output: the augmented event, $E$
Function AUGMENT($seed$, $B$)
  $E = \{seed\}$
  $\forall N$ s.t. $\exists M \in E, S(N, M)$
    if $C(N, B) > 0$ then
      if $C(N, B) > (C(N, E) + C(E, N))$ then
        $E = E \cup N$
  return $E$
```

Figure 4: AUGMENT Algorithm Pseudocode

nodes are likely to actually represent different events.[4] To avoid adding to an event a node that actually represents a different type of occurrence, we bias the cooccurrence function to penalize events that contain multiple nodes from the same narrative.

SPANIEL employs the AUGMENT algorithm, illustrated in Figure 4, each time it evaluates the addition of a new event to a sequence. The new event is seeded with an initial node, and AUGMENT then checks each node that is similar to the event. If the node occurs at least once with the neighboring event and does not occur more frequently with the event being augmented, the algorithm adds the node to the event. We show the pseudocode for augmentation at the beginning of a sequence; augmentation at the conclusion of a sequence is analogous, with $C(N, B)$ replaced by $C(B, N)$ in all cases.

Two special cases bootstrap event formation when a correlation of two events is first created. First, AUGMENT allows the InSeq criterion to be fulfilled by cooccurrence with a node *similar* to the existing neighbor, since that neighbor initially consists of a single node. Second, since that neighbor also previously had no context to constrain it, AUGMENT is also called on the neighbor.

3.2 The REVISE Algorithm

As previously mentioned, AUGMENT only has one neighbor against which to enforce the InSeq criterion. Fulfillment of the criterion using one neighbor does not ensure that the node will be acceptable given a second neighbor. Therefore, if a second neighbor to this event is added to the sequence, the REVISE algorithm removes nodes that fail to remain acceptable. Figure 5 illustrates the situation with a sequence containing three events, the new one having just been added at the front of the sequence. Each node in the new event meets the InSeq criterion, but now a node in its neighbor does not. REVISE removes such nodes from events. Any removals from the first event revised may also warrant removals from the next neighbor in the sequence and so on. Therefore, SPANIEL applies REVISE to each existing node in the sequence in order, starting with the event neighboring the new event.

[4]This does not mean that identical events cannot occur in a sequence.

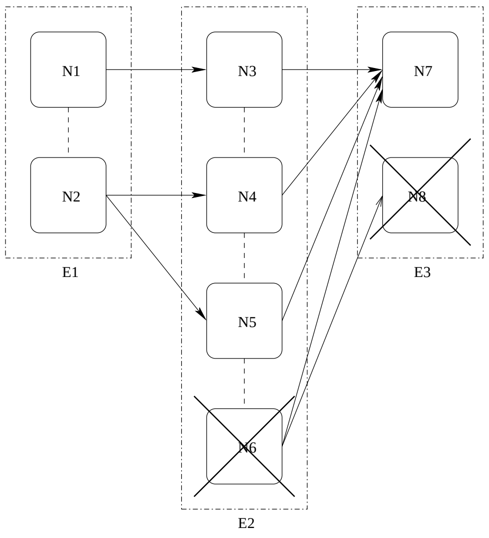

Figure 5: Example of an InSeq Violation

```
Input: An event sequence, $S = < E_1, E_2, \ldots, E_n >$
Output: An event sequence respecting InSeq
Function REVISE($S$)
  modified = true
  $i = 2$
  while (modified && $i \leq n$)
    modified = false
    $\forall X \in E_i$
    if $C(E_{i-1}, X) = 0$ then
      $E_i = E_i \setminus X$
      modified = true
    if $|E_i| = 0$ then
      return failure
    $i$++
  return $S = < E_1, \ldots, E_n >$
```

Figure 6: REVISE Algorithm Pseudocode

REVISE (see Figure 6 for the pseudocode) checks that all nodes in an event are still valid members, given the context provided by a new neighbor. If the algorithm removes any nodes from the event, it proceeds to check the next event in order. The pseudocode presents this ordering from left to right, as when an event is added to the front end of the correlation. The reverse traversal is analogous, with the loop proceeding from $n - 1$ down to 1 instead of from 2 up to n, where n is the length of the sequence. Note that the first (respectively last) event does not need to be checked, as AUGMENT ensures the InSeq criterion for this event with its single neighbor.

REVISE can have one additional effect. If revision results in any event having no nodes, the proposed event addition fails. In other words, the new event is inconsistent with the rest of the correlation.

In summary, the AUGMENT algorithm creates an event that contains all nodes that are similar to the original node and that cooccur with the neighboring event in the sequence. Since AUGMENT works on an event that has only one neighbor, REVISE checks these constraints when a second neighbor is added next to an event. Together, the algorithms enforce the two criteria we use to define events.

4 Evaluation

Automatic evaluation of individual events would be difficult, since they primarily have meaning in the context of a specific correlation. Therefore, we primarily employ manual evaluation. In addition, we describe a series of ablation tests that illustrate why events and our two event formation criteria are needed.

To conduct the evaluation, we applied SPANIEL to a corpus of narratives from the National Transportation Safety Board's online database of summaries of aviation accidents. This database, as of our experiments, contains more than 43,000 texts. We sampled five sets of approximately 4,300 texts each, and subdivided these sets with eighty percent of the sample used for training and twenty percent used for testing.

4.1 Event Cohesion

To evaluate event generation, we presented human judges with events acquired by our system and asked them to score each event by assessing the nodes' concurrence in describing an identifiable type of occurrence. The judges were asked to apply the following procedure:

1. Read all nodes in the event, and determine what occurrence the nodes describe.
2. Read all the nodes again, with the dominant concept of the occurrence in mind. For each node, assign a score of one if the node matches the concept or zero otherwise.
3. For each event, total the scores of the nodes. This gives a score for the event.

The first step is asks the judge to choose the dominant concept of each event based on their interpretation of the nodes. The second and third steps assign a score to each event based on the nodes it contains. We then divide this score by the number of nodes in the event to give a percentage of nodes that match the dominant concept of the occurrence represented by the event. We call these the *conforming nodes*.

We used two human judges, both of whom have backgrounds in computer science and aviation. Each judge was given the same set of results from the learning system, with each two event correlation separated into individual events. An example of an event, printed as the judges received it, is given in Figure 7. We generated correlations from the five different data sets, and blindly chose one set to give the judges. Figure 10 gives the distribution of the number of nodes in each event.

```
** Event 32 **
-> PILOT  MADE  FORCED LANDING
-> PILOT  PERFORMED  FORCED LANDING
-> PLT  MADE  LEFT TURN
-> PILOT  MADE  PRECAUTIONARY LANDING
-> PLT  MADE  FORCED LANDING
-> PILOT  LANDED ON GRAVEL BAR
-> PLT  APPROACHING_LAND IN OPEN FIELD
-> PLT  ATTEMPTED   LANDING
-> PILOT  ATTEMPTED_LAND AT NEARBY AIRPORT
-> PILOT  ELECTED_LAND IN FIELD
-> PILOT  LAND IN OPEN FIELD
-> PILOT  HAD_LAND_OFF   AIRPORT
-> PLT  MADE  EMERG LANDING
```

Figure 7: Example Learned Event

System	Judge 1	Judge 2
Baseline	0.623	0.696
Learned	0.925	0.930

Table 1: Evaluation Results

To provide a baseline, ten percent of the events presented to the judges were generated by probabilistically choosing a set of random nodes from the graph, with more frequent nodes chosen with higher probability. Each random event contains between two and ten nodes. The judges were not informed of this procedure, and thus we can obtain a measure of their scoring applied to a randomly generated event.

In Table 1, we give the mean fraction of conforming nodes for our event generation algorithm and the random baseline.[5]

We present more detailed results in two graphs, one for the results from each judge, in Figure 8 and Figure 9. We plot both the baseline and the learned results, giving the fraction of nodes that were judged conforming versus the size of the event in nodes. Where more than one event had the same number of nodes, we plot the mean conforming node fraction for the events. These graphs give an indication of the cohesiveness of the events as the size of the events increases. The cohesiveness of the learned events does not appear to drop dramatically as the size of the events increases, whereas the the baseline's performance does appear to decrease.

[5]Our second judge, instead of giving a numerical score and description of the event's dominant concept, simply put a question mark for each of the two baseline events of size 10. We interpret this result as no dominant concept, which equates to the existence of a pathological dominant concept of size one, and therefore one conforming node.

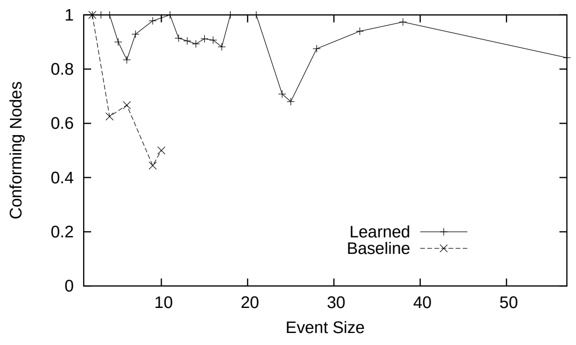

Figure 8: First Judge's Results

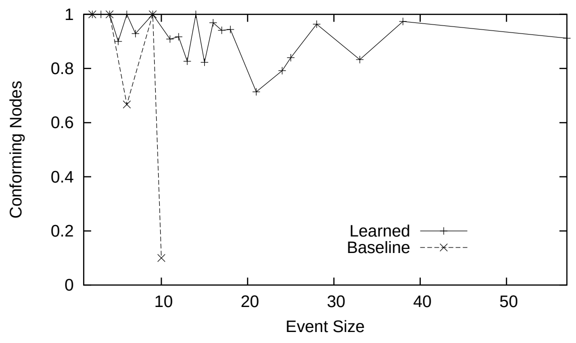

Figure 9: Second Judge's Results

4.2 Ablation Testing

We also evaluated the utility of events by eliminating SPANIEL's ability to acquire events. The ablated version of SPANIEL can only learn correlations of nodes, not events. We used the same five sets of training texts as input to the ablated system. The full system learned between 22 and 28 correlations meeting a probability threshold corresponding to approximately 12 occurrences in the training set, while the ablated system learned between zero and two correlations with the same probability.

We also ablated SPANIEL by removing each of the two criteria, Similarity and InSeq, individually. Removing either criteria rendered the system computationally infeasible to run. Neither version could expand even a single sequence in a reasonable amount of time. This phenomenon appears to be the result of two similar causes. If the InSeq criterion is disabled, the system encounters a chain reaction caused by similar nodes. Each time a node is added to an event, more nodes become similar to the event, causing uncontrolled expansion. If the Similarity criterion is disabled, any node that occurs in the proper order with the neighboring event can be added to the event. Since cooccurrence ignores intervening clauses, a potentially huge number of nodes can cooccur with an event, again causing uncontrolled expansion. These results support our hypothesis that, since event formation is an unsupervised process, it must be somewhat conservative to avoid allowing excessive noise into events.

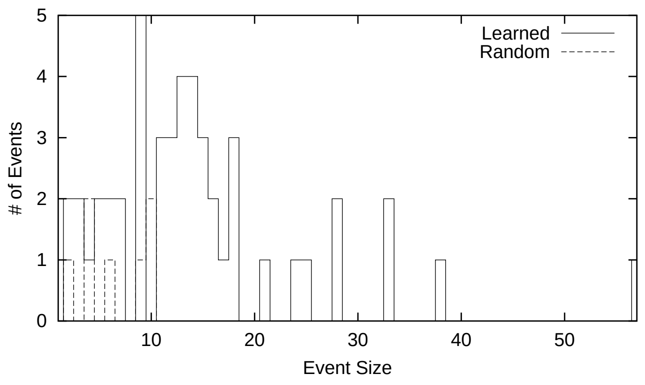

Figure 10: Event Sizes

5 Related Work

Two lines of research are relevant to this work. First, our research is based on research into scripts. Second, recent work in semantic lexicon learning is similar to our work, although it focuses on learning related words, not related clauses. In addition, extraction patterns and case frames bear some resemblance to our events. Riloff and Schmelzenbach's work (1998) is an example of this line of research. However, events use contextual information about other events, unlike extraction patterns and case frames.

The idea of a script originated with Schank and Abelson (1977) through research on human knowledge structures, and was demonstrated in the SAM system (Cullingford, 1978). Later work includes manual creation of a variety of knowledge structures including scripts to understand stories (Lehnert et al., 1983), application of manually generated scripts to the processing of newswire stories (DeJong, 1982), and a combination of applying manually generated scripts to information retrieval and applying genetic algorithms to adjusting existing scripts (Mauldin, 1989).

Semantic lexicons have been the focus of much research. WordNet (Fellbaum, 1998) is a prominent example of a manually generated lexicon. Two recent projects in learning semantic lexicons apply automated techniques to a small set of human provided seed words to create lists of words that the systems assign to the same semantic category. Each project uses a different technique to evaluate word similarity. Thelen (2002) uses similar context within a sentence. Phillips (2002) mines syntactic structures. Other researchers have also clustered words to create semantic lexicons. Lin (1998) created a thesaurus using syntactic relationships with other words.

Rooth et al. (1999) used clustering to create clusters similar to Levin verb classes (Levin, 1993). Pereira, Tishby and Lee (1993) clustered words according to context.

6 Conclusion

Both of our criteria play essential roles in the event generation process. Using similarity alone would combine all clauses that, while similar on the surface, actually are referring to different types of occurrences. Using context alone would combine all clauses encountered in the same position relative to some other clause. Either approach allows excessive noise to contaminate the resulting events.

Our event generation technique is an essential part of SPANIEL, our event correlation learning system. Without a means for combining nodes, the system would be unable to generalize between different authors' descriptions of occurrences, unless they used exactly the same terminology. While standardized lexicons have been investigated (Kamprath et al., 1998), their use has not become common. Therefore, competent event generation is required for success in detecting event correlations.

In addition to their role in event correlations, events could be used as information extraction tools. Using multiple fillers from different nodes for two primary slots makes the event potentially useful as a pattern for extracting fillers for the third slot from specific documents.

Our results show that this technique for generating events produces encouragingly coherent events and outperforms randomly grouping nodes. Our technique for exploiting the training texts during event generation is relatively simple. Possible future work includes testing other algorithms for combining nodes, such as standard clustering techniques. In summary, we have defined an interesting problem and provided useful insight into its solution, which furthers our research into learning event correlations.

7 Acknowledgements

We would like to thank Robert Cornell and Donald Jones for evaluating our system.

References

Peter Clark and Bruce Porter. 1995. Constructing scripts from components: Working note 6. Unpublished manuscript, University of Texas at Austin.

Richard Cullingford. 1978. *Script Application: Computer Understanding of Newspaper Stories*. Ph.D. thesis, Yale University, New Haven, Connecticut.

Gerald DeJong. 1982. An overview of the FRUMP system. In Wendy Lehnert and Martin Ringle, editors, *Strategies for Natural Language Processing*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Christiane Fellbaum, editor. 1998. *WordNet, An Electronic Lexical Database*. MIT Press.

Christine Kamprath, Eric Adolphson, Teruko Mitamura, and Eric Nyberg. 1998. Controlled language for multilingual document production: Experience with caterpillar technical english. In *Proceedings of the Second International Workshop on Controlled Language Applications (CLAW '98)*.

Wendy Lehnert, Michael Dyer, Peter Johnson, C J Yang, and Steve Harley. 1983. BORIS — an experiment in in-depth understanding of narratives. *Artificial Intelligence*, 20:15–62.

Beth Levin. 1993. *English Verb Classes and Alternations: A Preliminary Investigation*. The University of Chicago Press, Chicago, IL.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLINGACL '98*, Montreal, Canada, August.

Michael L. Mauldin. 1989. *Information Retrieval by Text Skimming*. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA.

Raymond Mooney and Gerald DeJong. 1985. Learning schemata for natural language processing. In *Proceedings of IJCAI-85*.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of english words. In *Proceedings of ACL-93*.

William Phillips and Ellen Riloff. 2002. Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In *Proceedings of EMNLP-2002*.

Ellen Riloff and Mark Schmelzenbach. 1998. An empirical approach to conceptual case frame acquisition. In *Proceedings of the Sixth Workshop on Very Large Corpora*.

Mats Rooth, Stefan Riezler, Detlaf Prescher, Glenn Carroll, and Franz Beil. 1999. Inducing a semantically annotated lexicon via em-based clustering. In *Proceedings of ACL-99*.

Roger Schank and Robert Abelson. 1977. *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Roger Schank and Christopher Riesbeck, editors. 1981. *Inside Computer Understanding: Five Programs Plus Miniatures*. Lawrence Erlbaum Associates, Hillsdale, New Jersey.

Michael Thelen and Ellen Riloff. 2002. A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In *Proceedings of EMNLP-2002*.