View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Texas A&M University

A GENERAL-PURPOSE APPROACH TO TEMPORAL EVENT ONTOLOGY CREATION

An Undergraduate Research Scholars Thesis

by

ALLISON BADGETT

Submitted to the Undergraduate Research Scholars program
Texas A&M University
in partial fulfillment of the requirements for the degree of

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Ruihong Huang

May 2017

Major: Computer Engineering

ABSTRACT

A General-Purpose Approach to Temporal Event Ontology Creation

Allison Badgett
Department of Computer Engineering
Texas A&M University

Research Advisor: Dr. Ruihong Huang
Department of Computer Science
Texas A&M University

One of the major challenges for modern data scientists is providing structure to data. Textual data is especially difficult to interpret and categorize. Much of the meaning found in this natural language data, like news articles or tweets, is contextual and potentially non-standard. Attempts have been made to manually create organizational ontologies, but this is usually limited to specialized sub-domains, as the task of providing a complete structure "by hand" across larger domains is unmanageable. We propose a general-purpose approach to event ontology creation, building upon a subevent classifier already developed in the initial stage of our research. In this work, we extract events from textual data and create a graph structure showing temporal relationships, using semantic and syntactic methods. This event ontology facilitates faster and more accurate automated data interpretation by providing a structure to textual data. The next stage in the "big data" phenomenon is not accumulating more data, but fully utilizing the vast amount of data already available. Event ontologies are a necessary step in this direction.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Ruihong Huang for her invaluable guidance throughout the development of this thesis, my colleague Wenlin Yao for providing his event noun list for use in this project, and the entire Texas A&M Natural Language Processing Group for their support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER I
INTRODUCTION AND LITERATURE REVIEW

1 Introduction

Textual data is difficult to process because of the complexities of natural language. There are an infinite number of grammatically correct text objects that can be created, making the implementation of definite pattern analysis techniques challenging. Additionally, much of the meaning of a word or sentence is not intrinsic to that item but rather derived from the context of the full textual element. Because of this, textual data must undergo preprocessing before a computer can adequately utilize it.

2 Literature Review

Ontologies offer a structure for natural language that can more easily be interpreted by a computer. They identify relationships between identified objects within the text, providing a structured semantic context. Ontologies are used in the fields of document classification, summarization and information extraction. Domain-specific, manually developed ontologies have already shown to be useful in areas like health surveillance. BioCaster is a web monitoring system that uses a specialized ontology to monitor textual data for information about health issues across the globe (Collier, et al. 2010). This system allows analysts to more quickly identify and synthesize media data about the health crises, as opposed to a manual search.

However, though ontologies have already proven their usefulness, ontology building is still an expensive process. Many ontology creation efforts are highly supervised, relying on human categorization and pattern matching to determine categories and item placement. These ontologies are often very application-specific and do not generalize well between domains (Astrova, et al. 2014). Other approaches tend to be too simplistic, relying on basic frequency matching to establish event relationships (Zhong, et al. 2010). More recently, there has been a movement toward less supervised approaches to ontology building, but these have focused on concrete object classification rather than more abstract event structuring (Balakrishna, et al. 2010).

3 Objective

We propose a semi-supervised learning approach that utilizes key contextual features in news articles to develop an ontology showing relationships between events in the text. The relationships of interest to us consist of the following: event hierarchy - for example, “scoring a goal” would be a subevent of the event “soccer game;” temporal ordering, which is relating events that typically occur in a time-based sequence; and similarity association, which is connecting events with very similar meanings.

CHAPTER II
METHODS

1 Overview

We use a multi-stage process to develop a temporal event ontology. In the first stage of the project, we provide order to the corpus. By using a very large corpus, we can have fairly low document-level recall and high corpus-level recall, which is why larger corpora are generally preferred for semi-supervised or unsupervised learning. Even if our classifier does not capture a relationship in one article, the relationship will likely occur multiple times throughout the corpus and be identified at least once.

2 Defining Temporal Relationships

First, we analyze temporal structure on the sentential level, focusing on event pairs. We define an earlier event as a precursor event and a later event as a post event within the context of this paper. Of course in some contexts, there may be multiple precursor events to the same post event and vice versa, as well as explicit concurrent relationships. In an article section describing a natural disaster, rising water temperatures and the development of a low pressure region may be the precursor events for a hurricane post event. Within this paper, we categorize each pair individually. So instead of identifying [rising water temperatures, low pressure region development $\rightarrow$ hurricane] as a temporal event relationship, we instead create pairs in the following manner: [rising water temperatures $\rightarrow$ hurricane], [low pressure region development $\rightarrow$ hurricane]. If it is clear within the context of the paper that the water temperatures rose concurrently with the low pressure region development, this will be indicated in the event graph structure.

Because of the focus on precision at this stage on the process, we avoid making assumptions about concurrency that are not well-supported syntactically. At these lower stages, making too many assumptions about temporal structure may boost recall, but negatively impact precision. Because we have the advantage of a large corpus, sentence- or even article- level recall is not a priority. However, imprecision at the article greatly affects the precision of the corpus itself. Rather than being mitigated by corpus size, imprecision is magnified as the article count increases.

3 Wikipedia Corpus

We chose to use the Wikipedia corpus, which is completely open-source for any use. Because temporal ordering heavily depends on the concept of narrative flow, we randomly selected about 1.5 million articles from the history category. These types of articles contain many time-transition phrases, such as "After [EVENT]..." and "During [EVENT]...", simplifying the process of temporal order. In addition to already possessing some sort of time-based structure, this class of article is very event-rich. They are very structured and condensed, with some explicit timestamps added to sections and sentences.

4 Dependency Parser

We use the Stanford Dependency Parser to determine word-level dependencies and part-of-speech tags on the sentence level. The part-of-speech tags originate from the Penn Treebank Project. For the input "This is a test of the dependency parser, showing its function." the parser output can be seen in Figure II.1.

Each word is grouped with its determined part of speech and linked with another word in the sentence via a dependency relation. For example, the word “showing” is linked to “function” by the “dobj”, or direct object, relation, indicating that “function” is the direct object of the verb “showing.” These dependencies can then be chained together to create more complex structures. Because the methods in this paper are highly dependent on this parser, any error at this stage will propagate through the subsequent levels. Because of this,

```
[((u'test', u'NN'), u'nsubj', (u'This', u'DT')),
((u'test', u'NN'), u'cop', (u'is', u'VBZ')),
((u'test', u'NN'), u'det', (u'a', u'DT')),
((u'test', u'NN'), u'nmod', (u'parser', u'NN')),
((u'parser', u'NN'), u'case', (u'of', u'IN')),
((u'parser', u'NN'), u'det', (u'the', u'DT')),
((u'parser', u'NN'), u'amod', (u'dependency', u'JJ')),
((u'test', u'NN'), u'advcl', (u'showing', u'VBG')),
((u'showing', u'VBG'), u'dobj', (u'function', u'NN')),
((u'function', u'NN'), u'nmod:poss', (u'its', u'PRP$'))]
```

Figure II.1: Sample dependency parser output

we primarily use structures that are parsed with high accuracy. For example, a dobj (direct object) relationship is more straightforward syntactically than an acl (clausal modifier of noun) relationship, and the parsing accuracy reflects this.

5 Data Structures

5.1 Event Graph Structure

To provide structure to the ontology, we used a graph structure. Each node contains an event, links to event nodes that temporally follow it, and links to event nodes that temporally precede it. Instead of functioning as a simple time line, the graph is multilevel, allowing for concurrent events. In the formation of the graph, we avoid making too many assumptions about the node ordering. For example, if we know that Event C occurs after Event B and that it also occurs after Event A, we cannot make any assumptions about the ordering of Event B and Event A. These events could be concurrent or one could precede the other. By using a multilevel structure, we do not have to make these structural assumptions. Event A and Event B can simply exist in their own levels, showing a relationship with Event C but not with each other as shown in Figure II.2.

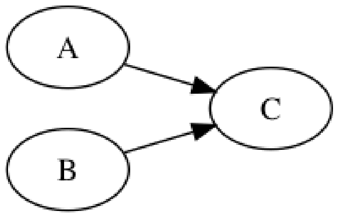


Figure II.2: Example graph layout

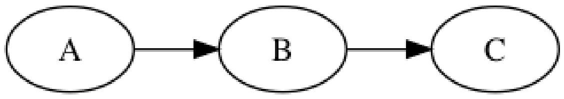


Figure II.3: Another example graph layout

Later on in our analysis, we may recognize a more explicit context in a different article that indicates that Event A precedes Event B in a case in which they both precede Event C. In this case, we can update this section of the graph accordingly as seen in Figure II.3.

5.2 Article Tree Organization

For the articles in the history category, approaching the full text as a unified whole posed difficulties to temporal ordering. Many of these articles are very long and contain multiple subsections that are not necessarily chronologically ordered. Lower-level article sections, like paragraphs and minor subsections, are more likely to be in temporal order than larger divisions that comprise multiple subsections. To mitigate this problem, we divided the articles into a tree structure with top-level sections separated into Wikipedia-defined subsections. The tree root holds the whole article, and the leaves are the paragraphs.

When forming the event graph for an article, subgraphs are formed at the leaf level and merged up the tree structure until a unified article graph is created at the root node. Further explanation of the merging procedure is detailed later in this chapter.

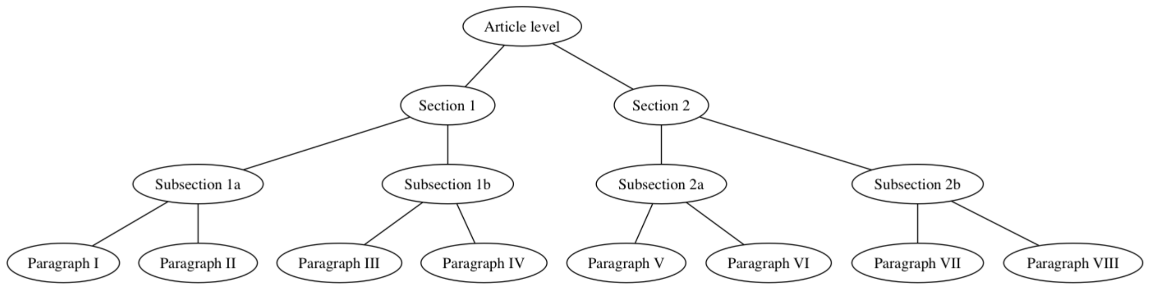


Figure II.4: Article tree layout

6 Event Extraction

6.1 Pattern-based Classification

We begin by identifying keywords and syntactic patterns within sentence contexts that indicate a temporal relationship between two events. Syntactically, events can be either simplified noun or verb phrases. For example, "[national] election" and "protested in [the main] square" are both valid event phrases as per our syntactic definition. The words within brackets indicate modifiers present in the sentential context but culled during the event phrase collection. This increases the generality of events, making them easier to link and cross-associate.

The patterns that we chose for event extraction do not represent all possible context constructions. Because of the versatility and sparsity of language, this is a near-impossible task. Instead, we selected patterns that we believe are dependable and produce an accurate result a high percentage of the time. These patterns are rarely affected by errors in the dependency parsing stage and have clear relationships to sentential events.

Because single-sentence context is the easiest to disambiguate, we began with the following patterns:

1. As a result of [precursor event], [post event].

2. Due to [precursor event], [post event].
3. After [precursor event], [post event].
4. Before [post event], [precursor event].
5. Until [post event], [precursor event].
6. ... [precursor event], and since then, [post event].
7. While [concurrent event], [concurrent event].

We also used the following consecutive-sentence context patterns:

1. ...[precursor event]. As a result, [post event].
2. ...[precursor event]. Since then, [post event].
3. ...[precursor event]. Consequently, [post event].
4. ...[concurrent event]. Meanwhile, [concurrent event].

Though temporal syntactic relationships across larger sentences contexts certainly exist, these relationship are relatively rare and difficult to disambiguate. For the purposes of this research, we did not attempt to identify explicit patterns across a larger context than two consecutive sentences.

6.2 Filtering the Event Set

We compiled a list of common phrases associated with some of the context patterns that may fit the syntactic pattern but do not contribute an event phrase. A partial listing can be found in Table II.1. For example, the following sentence fits one of the single-sentence patterns but does not contribute a precursor event:

After all, the volunteers had worked on a weekend.

After filtering out these phrases, we further narrow down our result set by improving the quality of noun phrases. The initial event noun set used was developed by Wenlin Yao, while a graduate student at Texas A&M University. At the time of publishing this thesis,

After all	After blood	Before the hour
After hours	After his heart	Since then
After the fact	Before evening	During the day

Table II.1: Subset of filtered idiomatic and common phrases

attack	operation	effort	study
talk	election	meeting	conflict
activity	negotiation	discussion	bombing
war	investigation	campaign	game

Table II.2: Subset of event noun set

his data has not yet been formally published. Only nouns from Yao's set are allowed to pass the initial stage. A sampling of noun phrases from his listing can be found in Table II.2. In total, there are 966 nouns in the original set.

A passive noun phrase identifies an object; for example, "the blue chair." An active phrase presents an event, such as "the revolution." To differentiate between these types of phrases, we use the lemmatization tool from the Stanford CoreNLP pipeline to identify the lemma of the core noun (Manning, et al. 2014). We then use the CoreNLP part-of-speech tagger to identify the part of speech of the lemma. If the lemma is determined to be a verb, we consider this noun to be active. If the lemma is not tagged as a verb, we then use a second test. We compiled a list of suffixes that indicate a process, such as '-tion,' and checked the list against the noun phrases that did not pass the first test. If the suffix is found, the phrase passes the second test. Phrases that do not pass at least one test are removed from the result set. This pipeline is detailed in Figure II.5.

Next, we attempt to improve the quality of the identified verb phrases. Much like noun phrases, verb phrases can be partitioned into two sets. For the purposes of this experiment, we focus on event verbs as opposed to state verbs. An event verb phrase indicates the

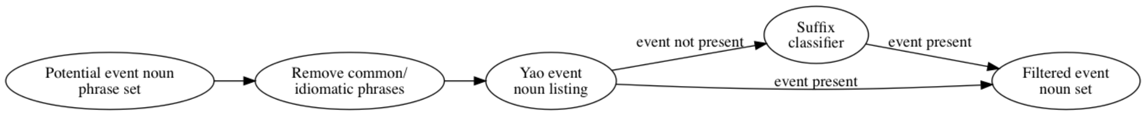


Figure II.5: Event noun phrase filtration pipeline

say	note	create	become
add	refer	suggest	know
use	ask	think	account
try	set	show	listen

Table II.3: Subset of state verb blacklist

occurrence of an action; for example, “fired the rocket.” A state verb describes the current state of a situation; for example, “concluded at this point that...” When part of the main verb phrase in a sentence, event and state verb are nearly indifferentiable from both a semantic and syntactic perspective. However, we realized that state verbs occur more frequently in participial constructions than event verbs. We did a frequency analysis of participial verbs throughout a subset of the corpus to create a “blacklist” of state verbs with a frequency cutoff at 95 instances. Any verb phrases containing one of these blacklisted verbs is removed from our result set. A subset of this blacklist is presented in Table II.3.

7 Graph Creation

7.1 Sentence-level Temporal Ordering

Using the article tree structure as defined in Section II.5.2, we create the event graphs for the leaf level - the paragraph level. We further subdivide this section into the sentence level and create a subgraph at this scope.

First, we extract the events from the main clause of the sentences and place them in an event list. Next, we compare the sentence against the list of patterns in Section II.6.1.

If the sentence matches one of these patterns, we extract the "outer" events, which can be precursor events, post events or concurrent events depending on the identified pattern. The graph construction process for each scenario is defined below.

1. **Precursor event identified**

   This is the most common pattern in the corpus, as it helps to maintain narrative flow. These patterns indicate an event that happened before the current narrative context. For example, in the following construction, the two main events in the sentence are within the “present” temporal context (even if they are past tense verbs), and the event in the pattern adjectival clause references a previous occurrence.

   After PRE_EVENT, A_EVENT and B_EVENT.

   To construct this graph, the main event phrases - A_EVENT and B_EVENT - are added to the graph between the START and END nodes. PRE_EVENT is then added to the graph in front of START to indicate that it precedes the whole clausal group. The resulting graph is found in Figure II.6.

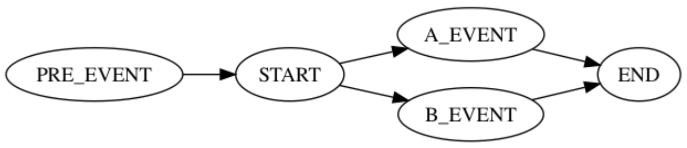


Figure II.6: Precursor event graph

2. **Post event identified**

   Post event constructions are less common than those of precursor events but serve an important narrative function. These patterns often provide a flashback to a recently discussed event. In this following example, the two post events in the adjectival clause occur in the present temporal context after the event in the main clause.

Before POST_EVENT_A and POST_EVENT_B, MAIN_EVENT.

To construct this graph, the outer events - POST_EVENT_A and POST_EVENT_B - are added between the START and END nodes. MAIN_EVENT is added before the START node. The construction of the post event graph is very similar to that of the precursor event pattern, but the position of the outer events and the main events are flipped. The resulting graph is shown in Figure II.7.

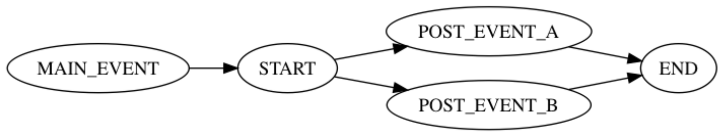


Figure II.7: Post event graph

3. **Concurrent event identified**

   Concurrent event patterns explicitly indicate that events are occurring simultaneously. Because we already make the assumption that all events in a sentence are occurring at the same time unless indicated otherwise by a post or precursor event pattern, this does not impact our graph structure. However, we chose to make this distinction anyway in case patterns developed in the future are more sophisticated. An example sentence is shown below:

   While CON_EVEN_A and CON_EVENT_B, MAIN_EVENT_A and MAIN_EVENT_B.

For this construction, all events in the sentence - both outer events and main events - are added to the graph between the START and END nodes. Figure II.8 shows the resulting graph.

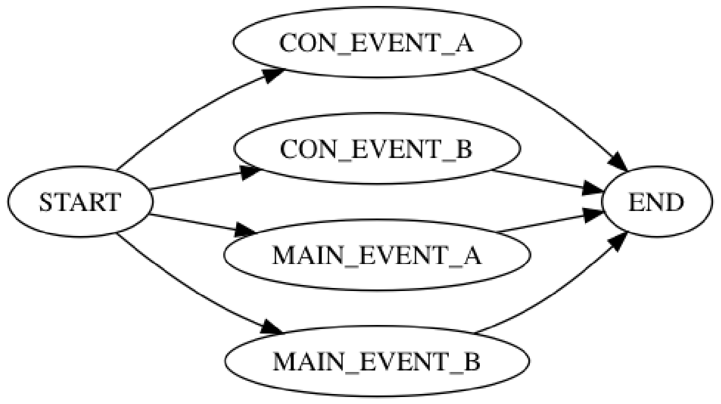


Figure II.8: Concurrent event graph

4. **No pattern identified**

   In the majority of sentences, none of our defined patterns fit. In this situation, we extract the event phrases within the main clause and add them to the graph between the START and END nodes, as shown in Figure II.9.

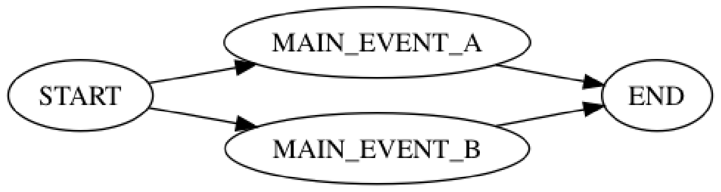


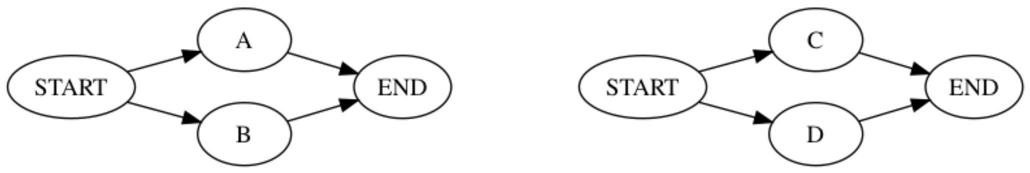
Figure II.9: No identified pattern graph

7.2 Higher-level Temporal Ordering

Once the sentence level subgraphs have been created, we merge them together at each node of the article tree. The merging procedure depends on the node configuration at the START and END nodes of the graphs to be merged. The merging procedures are outlined below. To maintain clarity, the first subgraph (earlier temporal and textual state) will be called Subgraph A and the consecutive subgraph with which it will merge will be called Subgraph B.

1. **Confluence of multiple nodes from both subgraphs**

   As shown in Figure II.10, in this configuration Subgraph A has multiple nodes feeding into its END node, and Subgraph B has multiple nodes branching out of its START node.



(a) Subgraph A (b) Subgraph B

Figure II.10: Procedure 1 - Subgraphs A and B before merging

   To merge the subgraphs, a new dummy node is created to bridge the gap. This maintains the temporal structure of the two subgraphs while still linking them. This construction is shown in Figure II.11.

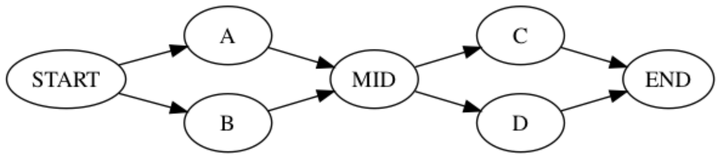
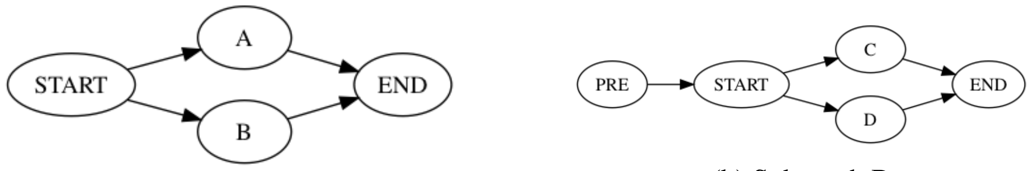


Figure II.11: Procedure 1 - Merged graph

2. **Confluence in which Subgraph B has a precursor node**

   In this construction, Subgraph B has a precursor node feeding into its START node as shown in Figure II.12.



(a) Subgraph A

(b) Subgraph B

Figure II.12: Procedure 2 - Subgraphs A and B before merging

In order to maintain the precursor node's temporal position, we insert a dummy node just as in the previous procedure.

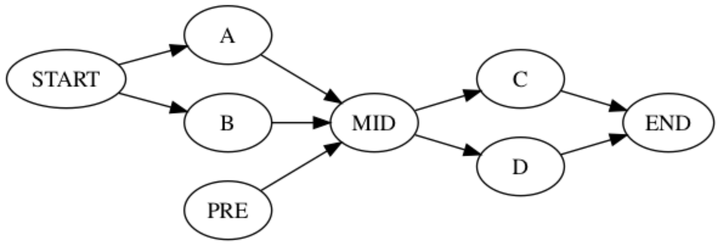
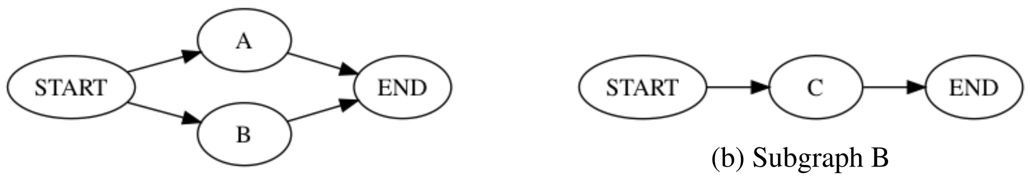


Figure II.13: Procedure 2 - Merged graph

3. **Confluence in which at least one subgraph only has a single connecting node**

   This case is the simplest configuration. At least one of the subgraphs has only a single node connected to the merge point. An example subgraph configuration is shown in Figure II.14.



(a) Subgraph A

(b) Subgraph B

Figure II.14: Procedure 3 - Subgraphs A and B before merging

   No dummy node is required, and the subgraphs can be linked directly.

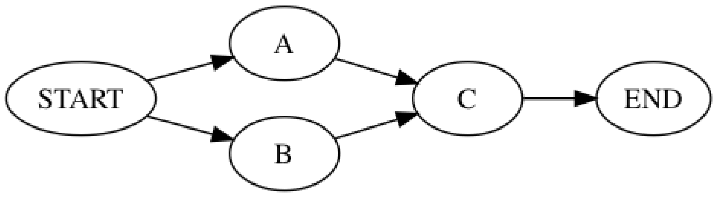


Figure II.15: Procedure 3 - Merged graph

CHAPTER III
RESULTS

1 Event Graph Visualization

Because graph structures can be hard to interpret and visualize solely through a debugging portal, we created a script that converts the graph to a text file with DOT graph description language. This interface allows the user to visualize the relationships between events in what can become a very large complex graph. All the graph images in this report are generated in this manner from real input data.

2 Event Graph Generation

To illustrate the graph generation process on the corpus itself, we use a paragraph from the Abraham Lincoln Wikipedia page:

> After the Union defeat at the First Battle of Bull Run and the retirement of the aged Winfield Scott in late 1861, Lincoln appointed Major General George McClellan general-in-chief of all the Union armies. McClellan, a young West Point graduate, railroad executive, and Pennsylvania Democrat, took several months to plan and attempt his Peninsula Campaign, longer than Lincoln wanted. The campaign's objective was to capture Richmond by moving the Army of the Potomac by boat to the peninsula and then overland to the Confederate capital. McClellan's repeated delays frustrated Lincoln and Congress, as did his position that no troops were needed to defend Washington. Lincoln insisted on holding some of McClellan's troops in defense of the capital; McClellan, who consistently overestimated the strength of Confederate troops, blamed this decision for the ultimate failure of the Peninsula

Campaign.

We divide the paragraph into sentences - considering a compound sentence divided by a semicolon as two discrete sentences. On the sentence level, we assume that all extracted events are consecutive unless the sentence fits one of the patterns outlined in Section II.6.1. We can see that the first sentence of this paragraph matches an identified pattern:

> After the Union defeat at the First Battle of Bull Run and the retirement of the aged Winfield Scott in late 1861, Lincoln appointed Major General George McClellan general-in-chief of all the Union armies.

After extracting the "main" event phrases from the main clause of the sentence, we extract the noun that is the direct object of "After" - "defeat". After extracting this noun, we identify significant modifiers. In this case, "defeat" and "Union" have a compound dependency relationship, indicating that the two words form a compound noun phrase - "Union defeat". The word "defeat" also has an adjectival prepositional phrase - "at the Battle". Though "Battle" has additional modifiers, they are not included. In order to maintain a level of generality in this stage, only direct modifiers are appended to the noun phrase. Once all events and modifiers are extracted, the graph is dynamically constructed. Through an edge traversal of the graph, the program outputs code in the DOT graph description language for automated graph construction. For this input, the program output is shown in Figure III.1. The associated graph is displayed in Figure III.2. The START and END nodes indicate the first and last nodes, respectively, of the graph. Because "Union defeat at Battle" is a precursor event to the entire main clause, this event precedes the START node, but is not technically the first node of the subgraph. The reason for this distinction will become clear when the subgraphs are merged.

Looking at the resulting graph, we notice that an event is missing - "retirement of Winifield Scott". This event should be placed in parallel with "Union defeat at Battle" as

```
digraph A {
        "Union defeat at Battle"->"START";
        "START"->"appointed general-in-chief";
        "appointed general-in-chief"->"END";
}
```

Figure III.1: Sentence DOT output

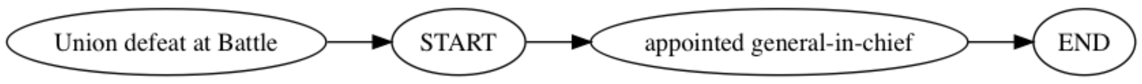


Figure III.2: Sentence-level graph

they are linked by a conjunction. In order to determine why the retirement event was not extracted alongside the defeat event, we consult the dependency relation list. The relevant subset of this listing in shown in Figure III.3. Rather than “defeat” and “retirement” being connected in a conjunction relationship, the parser erroneously conjuncts “Battle” and “retirement”. When parser errors occur, as in this case, this system is unable to correctly extract impacted events.

Next, the sentence-level subgraphs are merged into a paragraph-level graph following the procedures outlined in Section II.7.2. The resulting graph is shown in Figure III.4. Not every sentence is represented within the graph because not every sentence contains an event phrase. In the following sentence, no actual event is expressed - just a plan for future action within a passive construction; therefore, no event is added to the graph.

> The campaign’s objective was to capture Richmond by moving the Army of the Potomac by boat to the peninsula and then overland to the Confederate capital.

```
[((u'appointed', u'VBD'), u'nmod', (u'defeat', u'NN')),
((u'defeat', u'NN'), u'case', (u'After', u'IN')),
...
((u'defeat', u'NN'), u'nmod', (u'Battle', u'NNP')),
((u'Battle', u'NNP'), u'case', (u'at', u'IN')),
((u'Battle', u'NNP'), u'det', (u'the', u'DT')),
...
((u'Run', u'NNP'), u'case', (u'of', u'IN')),
((u'Run', u'NNP'), u'compound', (u'Bull', u'NNP')),
((u'Battle', u'NNP'), u'cc', (u'and', u'CC')),
((u'Battle', u'NNP'), u'conj', (u'retirement', u'NN')),
...]
```

Figure III.3: First sentence dependency parse subset

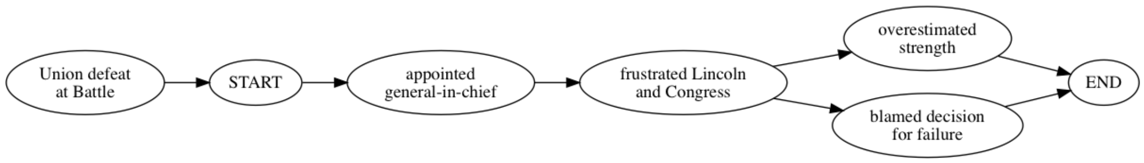

Union defeat at Battle
START
appointed general-in-chief
frustrated Lincoln and Congress
overestimated strength
blamed decision for failure
END

Figure III.4: Paragraph-level graph

Looking at the graph, we can see that our assumption that sentences would be listed in chronological order within a paragraph is justified. The event graph encompasses nearly all listed events (with the exception of the parser error) and organizes them correctly, recognizing complex temporal relationships.

CHAPTER IV
SUMMARY AND CONCLUSIONS

1 Contribution

This research is one of the first forays into building a general-purpose event ontology. Though large corpora of human-readable literature (e.g., Wikipedia) have been constructed, the development of computer-readable databases lags behind. This is due, in part, to the complexity of textual data. Some data is encoded syntactically; other data is semantic; other information relies on knowledge external to the text. Within this work, we developed a syntax-based method for extracting events from the text. Once these events were extracted, we placed them in a graph structure, showing article-level temporal relations.

This work has the potential to contribute to multiple areas of the language processing domain. By decomposing textual data into a graph structure, we build a base for article summarization. Disambiguation methods could also benefit from a temporal ontology. When human readers process a text, they interpret the data through a filter of external experience. For example, in the text below, the pronoun antecedents can be difficult for a computer to disambiguate:

> Michelle and Karen went skiing in the mountains. Michelle fell and broke her ankle. Karen helped her get back to camp. She checked into the hospital that evening.

In the last sentence, “she” could reasonably be either Michelle or Karen. Viewing the problem syntactically, a computer might select “Karen” as the antecedent because “Karen” was the most recently mentioned proper noun. However, this would not be correct. A human reader approaches this text with the understanding that a broken ankle often leads to a hospital visit. Through the development of the event ontology, we provide a similar knowledge base to be used in computational linguistics.

2 Further Study

2.1 Textual Timestamps

In historical article, dates and other temporal information is reference regularly. By developing a method to capture this information, graph development and merging procedures could be made more sophisticated, and cross-document comparisons could be made.

2.2 Expanded Pattern Extraction

Additional patterns could be added for event extraction and temporal context identification. Participial events could extracted and appropriately placed within the graph structure. This adds an extra level of complexity to ontology creation, as the temporal context of participial events is often more ambiguous than their main clause counterparts.

2.3 Event Generalization

In order to make comparisons between graph nodes, a generalization procedure is necessary. Though to a human reader, a “protest” is clearly a very similar event to a “demonstration,” it is difficult for a computer to attain this level of semantic understanding. By developing an algorithm to make these comparisons, the quality of this ontology would be greatly improved.

REFERENCES

[1] I. Astrova, A. Koschel, J. Lukanowksi, J. Martinez, V. Procenko, and M. Schaaf, “Ontologies for complex event processing,” *International Journal of Computer, Electrical, Control and Information Engineering*, vol. 8, no. 5, pp. 695–705, 2014.

[2] A. Badgett and R. Huang, “Extracting subevents via an effective two-phase approach,” *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 906–911, 2016.

[3] M. Balakrishna, D. I. Moldovan, M. Tatu, and M. Olteanu, “Semi-automatic domain ontology creation from text resources,” *LREC*, 2010.

[4] N. Collier, S. Doan, A. Kawtrakul, M. G. Reiko, A. Kawazoe, K. Takeuchi, ..., and D. Dien, “An ontology-driven system for detecting global health events,” *International Conference on Computational Linguistics*, pp. 215–222, 2010.

[5] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014.

[6] Z. Zhong, Z. Liu, W. Liu, Y. Guan, and J. Shan, “Event ontology and its evaluation,” *Journal of Information and Computational Science*, pp. 95–101, 2010.