Predicting an Effect Event from a New Cause Event Using a Semantic Web Based Abstraction Tree of Past Cause-Effect Event Pairs

Prajwal Shrestha
Department of Computer Science
University of Vermont, U.S.A.
pshrest1@uvm.edu

Byung Suk Lee
Department of Computer Science
University of Vermont, U.S.A.
bslee@uvm.edu

James P. Bagrow
Department of Mathematics and Statistics
University of Vermont, U.S.A.
jbagrow@uvm.edu

Abstract

Causation in its simplest form can be expressed as the pairing of a cause event with an effect event. Grouping and relating common past event pairs can be a key to predicting an effect event for a previously unseen cause event. This paper presents a project that (i) constructs an *abstraction tree* based on several semantic web datasets to represent all past events and then (ii) uses the abstraction tree to make such a prediction. To construct an abstraction tree, similarities between each pair of cause-effect events are measured using ConceptNet, DBpedia, and YAGO semantic web data and the measured similarity values are used by a clustering algorithm to group similar cause-effect event pairs together. To predict an effect event, the abstraction tree is traversed to find similar past cause events and then the paired effect events are extracted, which are then generalized using ConceptNet to form a new effect event. A user study shows favorable ratings on new predicted events compared with known baseline events.

1 Introduction

People have always been fascinated to know things about the future. Whether it is the prediction of election results or the weather, people have been intrigued to develop complex models that predict results accurately. These models are built on a large input data set, algorithms to process the data, and a knowledge base for inference. One such knowledge base is semantic web data. A semantic web contains the knowledge inherent in the data and is made of tiny pieces of data that are linked together so that computers can easily read them. One interesting application that can make use of such a knowledge base is found in the problem domain of event prediction (Radinsky et al., 2011, 2012; Radinsky and Davidovich, 2012; Radinsky and Horvitz, 2013; Dami et al., 2016).

This paper discusses a project *EEPAT (“Effect Event Prediction using an Abstraction Tree”)* under way to build and use such a model. The goal of the project is to organize all provided cause-effect (CE) event pairs so that they can be effectively used to predict an effect event for an arbitrary cause event. The model uses facts represented in semantic web data – specifically, ConceptNet (Luminoso, 2017), DBpedia (DBpedia Association, 2017), and YAGO (Max Planck Informatik Institut, 2017) – to build what we call an “abstraction tree” of the past CE event pairs. The tree is then used to predict an effect event given a previously unknown cause event that has occurred. An example is when ‘people smuggle drug’ then as a result ‘security is tightened’ or ‘security changes’. Information like this can be helpful to better understand the effect of the cause event or further take a countermeasure to prevent the predicted event from happening.

The CE event pair dataset used in this project is composed of 6009 news events from New York Times[1]. Approximately half of the events in the dataset are on topics related to violence, e.g., crimes, accidents, conflicts, crises, disasters. The knowledge base is limited to predicting effect events within this input data set.

This paper makes two key contributions. First, it presents a simple yet effective approach to predicting an effect event for a previously unknown cause event. The key idea is to construct an abstraction model of previously occurred causal relationships in the form of an abstraction tree (i.e., a hierarchy of CE event pairs clustered from the raw pairs) and then, given a previously unknown cause event, use the abstraction tree to find the best

[1] http://kiraradinsky.com/files/events_modelNYT.nt_partial.txt

matching top k CE event pairs and generalizing the effect events in these top k pairs to generate a predicted effect event.

Second, it validates the presented approach in terms of different implementation options, i.e., to (i) use the abstraction tree or not, (ii) compare the similarity between events "partially" or "totally" (for many events that do not have all of subject, object, and verb in their definitions), and (iii) predict a new effect event through exact single best match or abstracted multiple top k match.

In the remainder of the paper, related work is discussed in Section 2, the semantic web datasets are introduced in Section 3, the key concepts and approaches employed in the EEPAT project are presented in Section 4, the relevant implementation specifics are covered in Section 5, the evaluations and their results are discussed in Section 6, and a conclusion is made in Section 7.

2 Related Work

The work by Radinsky et al. (2012) and Radinsky and Davidovich (2012) had a direct influence on this EEPAT project. Their papers discuss the generalization of events using an abstraction tree and the creation of rules for predicting new future events from past events. Their "abstraction tree" is the same as ours in principle, albeit not in implementation. The scope and scale of their work were massive, with semantic data compiled from a dozen different huge ontologies and computations performed in a large-scale parallel platform. In contrast, our EEPAT project aims to build a system in a limited scope and scale within the available off-the-shelf computing resources and time and yet come up with acceptable results.

There are also other work that share some similarities to our work. Let us discuss some of the recent work published (Radinsky and Horvitz, 2013; Dami et al., 2016; Hu et al., 2017). Radinsky and Horvitz (2013) developed a model for predicting the probability a given effect event occurring at a given time. The model is extracted as a chain of events and expressed using the Bayes rule. Dami et al. (2016) used a Markov logic network model to represent events compactly using first-order logic and to enable probabilistic prediction of news events. Their primary focus was on event extraction from a news text rather than event prediction. Hu et al. (2017) considered an event-subevent hierarchy representation and used a model called the "context-aware hierarchical long short-term memory (CH-LSTM)" to predict a future subevent from a given past subevent. This model is not exactly a causal model, and they did not use semantic web data. None of them is using the approach characteristic of our work, i.e., predicting an effect event for a given cause event using a causal relationships model built using semantic web data.

3 Semantic Web Datasets

This project is driven by knowledge represented by semantic web data. Currently we focus on three datasets – ConceptNet, DBpedia, and YAGO – which altogether cover different aspects of semantic knowledge base comprehensively. DBpedia and YAGO are connected to each other, and ConceptNet is partially connected to DBpedia. They are searched separately in our work.

ConceptNet is a semantic network that has information about basic *common sense* knowledge or very simple facts – the kind of information required to display basic human intelligence. It contains mostly information about the words or phrases we commonly use to state facts. Some examples are 'lighting match causes fire', 'fire related to hot', and 'dog capable of barking'. Although ConceptNet is a multilingual semantic dataset, we use only English words or phrases in the project.

The basic unit of knowledge in ConceptNet is an *edge*. It has the start term, the end term, and the relation between them. For example, 'lighting match causes fire' is represented as an edge {start_term: /c/en/lighting_match, rel: /r/cause, end_term: /c/en/fire}.

DBpedia is a knowledge base retrieved from the Wikipedia. Since Wikipedia stores data related to specific people, places, countries, movies, incidents, and organizations, it makes DBpedia a powerful dataset that complements ConceptNet dataset, which focuses on common sense knowledge base. All structured contents and metadata about Wikipedia page are organized and stored in DBpedia. We can read about something in Wikipedia, whereas we query the same information in DBpedia.

YAGO extracts information from WordNet and GeoNames datasets as well as Wikipedia. Unlike DBpedia, YAGO focuses more on organizing data taxonomically and on having more validation of the facts it stores.[2] It has only 109 distinct predi-

[2] "The accuracy of YAGO has been manually evaluated, prov-

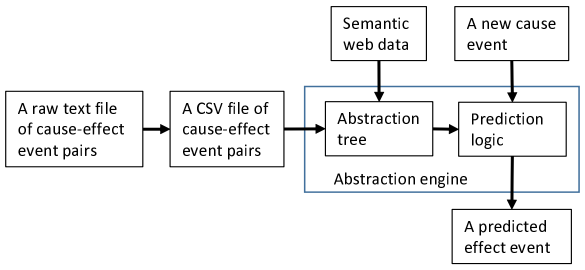


Figure 1: A high level overview of the system.

cates to relate each resource to another. Therefore, it is more specific to the subject than DBpedia, where DBpedia stores a lot of metadata information about the Wikipedia page itself. YAGO also stores additional information about the time and space of each fact.

Every data in a semantic web is a *resource* in its namespace. Each resource is represented by a unique URI and is linked to another resource via a *predicate*, also referred to as a *property* of a resource. In a semantics web, a resource is represented by a *node* and a predicate is represented as a directed edge.

4 EEPAT Concepts and Approaches

The main task in this project is to build an abstraction engine that is capable of predicting new effect events for a new cause event. The engine comprises an abstraction tree and the built-in prediction logic. The abstraction tree consists of past CE event pairs that are clustered using the knowledge from semantic web data. Figure 1 shows a high-level overview of the system.

The construction of an abstraction engine is dependent on the specifics of event representation and the semantic similarity measure used to cluster CE event pairs.

4.1 Event Representation

An event e is defined as a triple consisting of a subject URI (s), an object URI (o) and a verb URI (v). Each subject, object and verb is identified with a unique URI in semantic web data. That is, given the set U of all URI's, $e = \langle s, o, v \rangle$ where $s, o, v \in U$. Further, a *cause-effect (CE) event pair* $\langle c, e \rangle$ represents the fact that the event c causes the event e.

4.2 Calculating Similarity between Resources and Events

Events are represented by resources. Therefore, calculating similarity between resources leads to calculating similarity between events, which in turn leads to similarity between CE event pairs.

4.2.1 Similarity between Resources

In this project similarity between resources r_1 and r_2, $\sigma(r_1, r_2)$, is calculated using Jaccard similarity as

$$\sigma(r_1, r_2) = \frac{|P_{r_1} \cap P_{r_2}|}{|P_{r_1} \cup P_{r_2}|}$$

where P_{r_i} ($i = 1, 2$) is the set of the property-resource pairs, (p_j, r_j), such that r_i is related to r_j ($r_j \neq r_i$) via p_j. The rationale for this measure is that if two resources are similar, then they are bound to have common resources shared through the same relation between them. For example, if 'table IsA furniture' and 'chair IsA furniture', then 'table' and 'chair' are similar in that they are related to the same object 'furniture' through the same relation 'IsA'. The higher the number of common objects, the higher the similarity is. Note that Jaccard similarity is a "groupwise" measure of topological similarity, which is known to be suitable for ontological instances like resources (Wikipedia, 2017).

4.2.2 Similarity between Events

Since events are composed of resources $\langle s, o, v \rangle$, similarity between events, σ_E, is calculated by taking an average of the similarities between each corresponding pairs of resources. That is, given two events $a = \langle s_a, o_a, v_a \rangle$ and $b = \langle s_b, o_b, v_b \rangle$, s_a is compared with s_b, o_a with o_b, and v_a with v_b for similarity calculation between a and b.

Most of the events do not have all three terms (i.e., subject, object, verb), and their implications on the calculated similarity measure differ. In this project, we use two different approaches to address it. An event may be partial with some of the resources missing or total with none missing. The first approach accommodates all events whether they are partial or total by calculating the arithmetic average of only those terms that exist, hence with non-zero σ values. (This approach is the same as that used by Radinsky et al. (2012).)

$$\sigma_E(a, b) = \frac{\sigma(s_a, s_b) + \sigma(o_a, o_b) + \sigma(v_a, v_b)}{\text{number of non-zero } \sigma \text{ terms}}$$

The second approach gives preference to an event with all three types of resources by always dividing by 3, the number of resources needed to completely represent an event.

$$\sigma_E(a, b) = \frac{\sigma(s_a, s_b) + \sigma(o_a, o_b) + \sigma(v_a, v_b)}{3}$$

ing a confirmed accuracy of 95%." (Max Planck Informatik Institut, 2017)

With this approach, evidently similarity between events having all three resources is greater than similarity between events with only one or two resources present. We call the first approach the *partial event similarity* and the second approach the *total event similarity*.

4.2.3 Similarity between Cause-Effect Event Pairs

Similarity between CE event pairs, σ_{CE}, is calculated as an average of the similarity between the cause events and the similarity between the effect events. That is,

$$\sigma_{CE}(\langle c_i, e_i\rangle, \langle c_j, e_j\rangle) = \frac{\sigma_E(c_i, c_j) + \sigma_E(e_i, e_j)}{2}$$

Note that dissimilarity, δ_{CE}, is

$$\delta_{CE}(\langle c_i, e_i\rangle, \langle c_j, e_j\rangle) = 1 - \sigma_{CE}(\langle c_i, e_i\rangle, \langle c_j, e_j\rangle)$$

4.3 Construction of Abstraction Tree

In a large dataset, it is computationally expensive to look up all past CE event pairs for processing. Thus, event pairs need to be organized and stored in a data structure that enables efficient retrieval and storage.

The abstraction tree represents a hierarchical grouping of semantically similar past CE event pairs. It acts as a knowledge source of past CE events, which is used to predict new effect events. The tree is constructed by clustering all the input CE event pairs using the hierarchical agglomerative clustering (HAC) algorithm. As HAC uses a bottom-up approach, each observation (i.e., each CE event pair) starts in its own cluster. Clusters are merged as they move up the hierarchy. The algorithm calculates the similarity between every CE event pairs to group the most similar event pairs into one cluster.

The objective of constructing the abstraction tree is to come up with an abstract (or generalized) event pair that represents other event pairs in a cluster. The CE event pair representing the root of the tree is the most generalized event pair that represents all event pairs in the input data set. As we traverse down the tree, we find increasingly more specific event pairs and as we traverse up, find increasingly more general event pairs.

Each node in the resulting tree corresponds to a cluster of CE event pairs that are semantically similar. Each cluster is represented by a clustroid, which is the event pair closest to the centroid of a cluster. In other words, a clustroid acts as the abstract CE event pair representing all CE event pairs in the same cluster and is stored as a node in the abstraction tree.

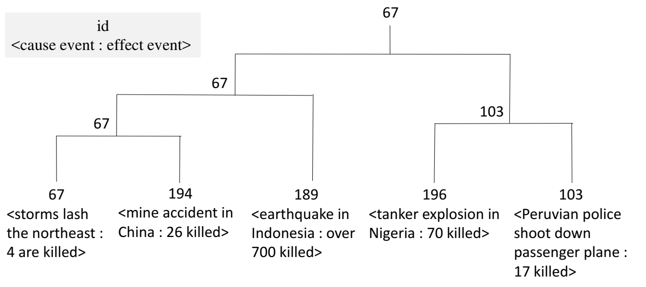

id
<cause event : effect event>
67
67
67
103
67
<storms lash the northeast : 4 are killed>
194
<mine accident in China : 26 killed>
189
<earthquake in Indonesia : over 700 killed>
196
<tanker explosion in Nigeria : 70 killed>
103
<Peruvian police shoot down passenger plane : 17 killed>

Figure 2: An abstraction tree constructed from five input cause-effect event pairs.

In the illustration shown in Figure 2, five actual CE event pairs selected from the New York Times data set[3] were clustered together to construct an abstraction tree. Initially, each CE event pair is a cluster of its own, containing only one node. The hierarchical clustering of n CE event pairs generates $2n - 1$ clusters across the hierarchy and, hence, there were nine clusters generated in this example. Clustroids were chosen as shown at each level of the hierarchy. Note that the clustroid may change as the bottom-up clustering progresses toward the root.

When merging two clusters to form a new cluster, one CE event pair is selected as the clustroid out of all CE event pairs in the new cluster. It finds the pair whose dissimiliarity from all the other pairs in the same cluster is the minimum. The selection of a clustroid is based on the majority ruling among the following three measures:

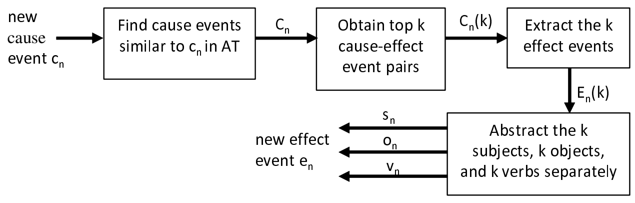
- Sum of the distances to the other nodes in the same cluster
- Sum of squared distances to the other nodes in the same cluster
- Maximum distance to the other nodes in the same cluster

Using the above three measures, a clustroid is selected based on the following mechanism:

1. The node that minimizes the largest number of these measures is selected as the clustroid.
2. If there is no majority node, then select the node that minimizes the sum of squared distance measure.

In this project the HAC algorithm was implemented by modifying the code obtained from the SAPE ("Software and Programmer Efficiency") Research Group (2015). The run time complexity of the HAC algorithm is $O(n^3)$, which is quite

[3] See footnote 1.


new cause event c_n
Find cause events similar to c_n in AT
C_n
Obtain top k cause-effect event pairs
$C_n(k)$
Extract the k effect events
$E_n(k)$
Abstract the k subjects, k objects, and k verbs separately
s_n
o_n
v_n
new effect event e_n
Figure 3: Effect event prediction workflow.

expensive for disk-resident execution of the algorithm for the given dataset size (about 6000 CE event pairs). Therefore, all the event pair data were loaded into main memory for efficient execution and the constructed tree was saved to a disk file for future reuse.

4.4 Prediction using the Abstraction Tree

The prediction of an effect event for a new cause event is made based on the constructed abstraction tree. That is, the abstraction tree serves as a knowledge base for finding the related cause events and its corresponding effect events. Note that, since the abstraction tree construction uses *semantic webs* to measure the distance between CE event pairs, the parent node can be considered a semantic generalization of a child node. While this may not be always agreed between all parent-child nodes, regardless the parent node is still an effective representation of all its children nodes in terms of the semantic distance, for a given a new input cause event. Besides, in order to identify the node (i.e., CE event pair) whose cause event matches the input cause event most closely, it is always more efficient to search down the abstraction tree than to scan all leaf nodes directly.

The abstraction tree, saved in a disk (see Section 4.3), is loaded into the memory so that it can be traversed quickly and efficiently for prediction.

There are two main steps in predicting a new effect event, as shown in Figure 3. First, for a new cause event (c_n), a candidate set of related past cause events ($C_n(k)$) are retrieved from the abstraction tree (AT). Second, the effect events ($E_n(k)$) that are paired with these past cause events are abstracted to form a new effect event ($e_n = \langle s_n, o_n, v_n \rangle$).

4.4.1 Finding a Candidate Set of Cause-Effect Event Pairs

The abstraction tree is traversed to find the past CE event pairs whose cause effects are similar to the given new cause event. The found event pairs are included in the *candidate set* for prediction. Note that while searching the abstraction tree for past event pairs, only the event pairs whose cause events are similar enough (i.e., more than a threshold) to the new cause event are included in the candidate set. This helps to retrieve only the past CE event pairs that are more relevant to a new cause event.

Specifically, while searching the abstraction tree down starting from the root node, for each node (i.e., CE event pair) visited, the similarity between the cause event in the pair and the given new cause event is calculated as the similarity between the node and the new cause event. Then, a node is included as a candidate only when its similarity to the new cause event is greater than the similarity of at least one of its children nodes to the new cause event. The search then traverses to each child node whose similarity to the new cause event is greater than the current node. The recursive traversal stops when none of the children nodes has similarity greater than the current node. This strategy helps to avoid including over-generalized events from the upper layers of the tree or over-specific events from the lower layers of the tree.

4.4.2 Predicting an Effect Event from the Candidate Set

The effect events in the selected candidate event pairs are then abstracted to construct a new effect event. We employed two different approaches, discussed below.

Exact Single Match

This approach returns the effect event of the one cause event that is the closest match to the new cause event. That is, out of all CE event pairs in CE_n, the event pair $\langle c_h, e_h \rangle$ whose c_h has the highest similarity to c_n is selected as the best match and, then, e_h is returned as the predicted effect event for the new cause event c_n. Note this is not a new predicted effect event but an effect event that has already occurred in the past. If the new cause event has already occurred in the past, then this method returns the most plausible event. Besides, this effect event is helpful as the baseline to compare effect events generated from the prediction approach.

Abstracted Multiple Match

In this approach, top k CE event pairs that most closely match the new cause event are considered. Effect events in these top k event pairs are abstracted to construct a new effect event. Abstraction of event pairs is based on the ConceptNet semantic web dataset. The set of subjects, the set of

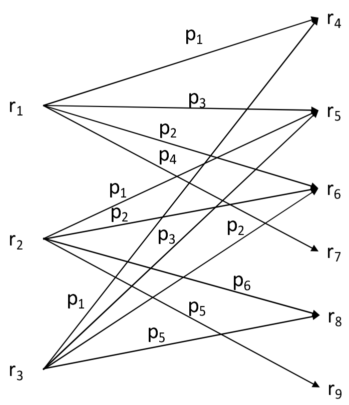


Figure 4: An example adjacent resources of r_1, r_2 and r_3.

objects, and the set of verbs from the effect events in the top k event pairs are abstracted *separately* to create a new effect event with the resulting subject, object, and verb. Since this effect event is generated from multiple CE event pairs, it is likely to give accurate and sensible results.

The abstraction of a set of resources finds the *most common adjacent resource*, which has a common predicate from the largest number of resources in the set. For example, as shown in Figure 4, given three resources r_1, r_2, and r_3 with the predicates $p_1, p_2, ..., p_6$ to other resources, *abstract*($\{r_1, r_2, r_3\}$) returns r_6 as the most common adjacent resource of r_1, r_2, and r_3 because it is connected from all three of them (via the common predicate p_2).

Two alternative abstraction methods have been employed: *Exclusion Set (ES)* and *Ranked Inclusion Set (RIS)*. In the ES method, those relations from ConceptNet that do not help with the abstraction process are excluded from consideration. Seven relations were excluded as a result. They are '/r/HasContext', '/r/Antonym', '/r/RelatedTo', '/r/part of', '/r/HasPrerequisite', '/r/CapableOf', and '/r/FormOf'. One of them is the antonym relation ('/r/Antonym'), and the rest are references to other resources. None of them can be used for generalization abstraction. In the RIS method, relations are ranked and processed in the order of their priorities. Thus, a relation with a lower priority is taken into consideration only if none of the higher priority relations is found during the abstraction. In our implementation, the resulting ranked relations, from the highest priority first, were '/r/Isa', '/r/Synonym', '/r/DerivedFrom', and '/r/DefinedAs'.

5 Implementation Backdrops

5.1 Platforms and Tools Used

As this project deals with large semantic web data sets, platform setup is a critical part of the entire project. The server was set up so that it could process queries on a large data set efficiently through the Fuseki web server.

The project uses Apache Jena Framework to store imported semantic web data. TDB persistent storage and Fuseki server, which are part of the Apache Jena framework, are used. Specifically, TDB is used to store DBpedia and YAGO semantic web data, and Fuseki Server has been set up to access semantic web data using SPARQL queries through the HTTP protocol. In addition, the project uses MongoDB to store application data. MongoDB is an easy-to-use, popular NoSQL database and is the main backend database to store the data retrieved as the result of a query. The data retrieved from the semantic web are stored in this database for frequent accesses.

Java Spring Boot, which is a rapid application development platform, was used to build the project codes and Maven tool was used to manage the dependency among project modules. In addition, Python was used to parse the input data file consisting of 6009 CE event pairs and to write the output CE event pairs as a CSV file (see Section 5.2).

The computer used in the project had 3.4 GHz quadcore CPU with 16GB RAM and 1TB SSD disk space. Both the database server and the application server were running on the same computer.

5.2 Preparation of Input Cause-Effect Event Pairs for Abstraction Tree Construction

The text input data file consists of the 6009 CE event pairs extracted from the New York Times website and formatted to be parsable so that a causal connector separates between a cause event and an effect event within a CE event pair, and each event is described as the triple of subject, object and verb. (As mentioned earlier, many events have some elements of the triple missing.)

There are seven distinct causal connectors in the input data file. They are listed in Figure 5 along with their number of occurrences in the dataset. Among them, the causal connector 'despite' was not considered because it implies the opposite, that is, $\langle$Event_A, despite, Event_B$\rangle$ indicates 'Event_B should not cause Event_A'. Consequently, 5504 textual CE event pairs were consid-

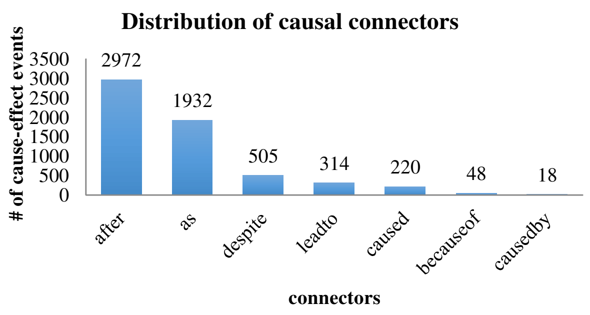


Figure 5: Distribution of causal connectors in the data set of the input file.

Event Pairs	**Cause Event**	**Effect Event**
<Event_A connector: leadto Event_B>	Event_A	Event_B
<Event_A connector: caused Event_B>	Event_A	Event_B
<Event_A connector: as Event_B>	Event_B	Event_A
<Event_A connector: becauseof Event_B>	Event_B	Event_A
<Event_A connector: causedby Event_B>	Event_B	Event_A
<Event_A connector: after Event_B>	Event_B	Event_A

Table 1: Causal connectors used in the project.

ered out of the 6009 pairs in the input file. Table 1 summarizes how the remaining six causal connectors were used to identify the cause event and the effect event.

Using the clues of causal connectors above, a Python script parses the input text file to generate an output as a CSV file. Each entry in the output CSV file is the pair of a cause event and an effect event, each of which is represented as the triples of a subject resource, an object resource, and a verb resource for each cause event and effect event. The output CSV file in this project consists of 5912 distinct resources (subject/object/verb), of which 1602 resources are verbs and 4310 resources are subjects or objects.

5.3 Using Semantic Web Data and Mapping Resources to URI

This project uses semantic data to find similarity between two independent events, which drives the construction of an abstraction tree. As events are composed of resources that are mapped as URIs in a semantic web, the semantic web data for a given resource of an event is easily accessible through its URI. As mentioned earlier, this project uses three semantic web datasets – DBpedia, ConceptNet, and YAGO. Each resource must be mapped to a unique URI before data can be retrieved from the semantic web.

As each event in the input data set of CE pairs is represented in the form of a DBpedia URI, DBpedia is used as a reference dataset to map URIs. Any given subject, object or verb is first mapped

Similarity measure	**Prediction method**	**Source of candidate set**
Partial event similarity	Exact Single Match	Abstraction tree
Total event similarity	Exclusion Set Based Abstracted Multiple Match	Raw cause-effect event pairs
	Ranked Inclusion Set Based Abstracted Multiple Match	

Table 2: Organization of the experiments

to a URI with the help of this dataset. DBpedia also provides lookup service to retrieve URI for an object that does not have a URI.

ConceptNet is the primary source of knowledge base for the construction of an abstraction tree. Since most of the input resources representing events are common nouns or verbs, ConceptNet is an ideal knowledge base to represent a common sense knowledge graph. A DBpedia URI is replaced by a ConceptNet prefix string to generate a ConceptNet URI. For example, http://dbpedia.org/resource/Swing is mapped to http://api.conceptnet.io/c/en/swing.

In this project, YAGO complements ConceptNet knowledge base. YAGO is derived from the Wikipedia knowledge base and, therefore, has dense knowledge resources related to people, locations, and events that are not available in ConceptNet. Thus, if a resource is not found in ConceptNet, then YAGO dataset is fetched to extract data. For this, a DBpedia URI is replaced by a YAGO prefix string to generate a YAGO URI. For example, http://dbpedia.org/resource/Larry_Davis is mapped to http://www.yago-knowledge.org/resource/Larry_Davis.

6 Evaluations

6.1 Experiment Design

Table 2 summarizes the setup of the experiments. It is centered on the two aspects discussed in Section 4 – the similarity measure between CE event pairs and the prediction method for generating an effect event. On top of that, another aspect has been added to compare the cases of whether an abstraction tree is used or not to find a candidate set. The latter case provides a baseline performance because it generates a candidate set by scanning all the raw input CE event pairs instead of traversing down the abstraction tree of them.

Based on this setup, 12 predicted effect events were generated for each input cause event. The results were first examined manually by interpreting the output effect events based on common sense. Then, a survey was conducted for a user study, where a group of people were asked to give scores ranging from 1 to 10 for each predicted effect event, with 1 being the least likely and 10 being the most likely predicted effect event for a given input cause event.

6.2 Experiment Results

Ten input cause events were manually built using a list of the top 20 most frequently occurring subjects, objects, and verbs. Part of the output results obtained from these input cause events are shown in Table 3 and Table 4. For each of the ten cause events (i.e., rows), the results were generated for each combination (i.e., column) of the similarity measure, the candidate set retrieval source and the prediction method summarized in Section 6.1. Table 3 shows the results from using the partial event similarity measure, where only the CE event pairs having similarity of at least 0.6 were considered. Table 4 shows the results from using the total event similarity measure, where only the CE event pairs having similarity of at least 0.3 were considered.

Based on the results above, a survey was conducted to validate and compare the performances of the event similarity measures, the abstraction tree, and the prediction methods. 41 people participated in the survey and, for each input cause event, gave scores for each of the 12 predicted effect events. Table 5 shows a statistical summary of the ratio of numeric scores.

6.3 Discussion of the Experiment Results

The focus of discussion is on answering the following three questions.

- Did the abstraction tree (AT) produce better results than the baseline (i.e., looking up the entire set of CE event pairs)?
- Which similarity measure resulted in better prediction between the partial and the total event similarities?
- Which prediction method yielded better results between the Exclusion Set (ES) and the Ranked Inclusion Set (RIS), and how do they compare with the baseline Exact Single Match (EMS)?

Abstraction Tree

When the results in Table 3 and Table 4 are manually checked, the results predicted by the abstraction tree (AT) are arguably more generalized and accurate while may not be conspicuous enough in all cases. The survey result indicate that predicted effect events from the AT are better than the baseline that does not use the AT by about 16%. This ratio may not seem significantly large, but note that using the AT is a model-driven approach (as opposed to an instance-driven approach) and, therefore, inherently carries the risk of losing the specificity due to model abstraction. Therefore, the fact that the results obtained from the AT are better than the other approach demonstrates that clustering of CE event pairs by the HAC algorithm is truly effective.

The AT is thus important to the capability of predicting an effect event for a given new input cause event. Note that not using the AT cannot find a generalized CE event pair in the same way it is possible using the AT. Moreover, the approach not using the AT will not scale as the AT approach will, as searching the raw input CE event pairs takes linear time with the number of CE event pairs whereas logarithmic in the AT approach.

Event Similarity Measure

The results in Table 3 and Table 4 are completely different. Let us consider the new input cause event 'Storm cause Fire' as an example. In Table 3 (i.e., partial similarity), the verb 'cause' plays a dominant role, that is, the output is based on the cause events that only match the verb 'cause' because it results in higher similarity. In contrast, the same event in Table 4 (i.e., total similarity) tries to match events with all the resources present. Since the three resources subject, object, and verb altogether represent an event, the partial similarity fails to deliver good results whereas the total similarity, which favors events that have all three resources in place, helps to improve the quality of the abstracted effect event. Consequently, the total similarity approach yields far better results. This is also supported by the results from the survey done for user study, where the total event similarity showed about 76% higher score than the partial event similarity.

Event Prediction Method

The scores between ES and RIS are comparable, with RIS slightly (i.e., 4%) better than ES. Both ES and RIS abstract a set of resources to create a generalized effect event and evidently these two yield similar results in terms of the apparent plausibility of the predicted effect events. On hindsight, the reason for these similar results is that the actual relationships followed for the abstraction are common between ES and RIS. Their results used interchangeable words to represent similar events in different forms, for example 'citi-

			Predicted effect events from all cause-effect event pairs			Predicted effect event from the abstraction tree		
#	Resource	Input cause event	Exact single match	Exclusion set based abstracted multiple match	Ranked inclusion set based abstracted multiple match	Exact single match	Exclusion set based abstracted multiple match	Ranked inclusion set based abstracted multiple match
1	Sub	Storm		wrestler	wrestler		wrestler	wrestler
	Vrb	cause	discount	think	think	rule out	think	think
	Obj	Fire	Sabotage	viral_infection	viral_infection	Probe	viral_infection	viral_infection
2	Sub	Police_officer	Group	set	set	Group	set	set
	Vrb	kill	suspend	change	change	suspend	expel	expel
	Obj	Gun	Work	medium	medium	Work	oeuvre	human_activity
3	Sub	Gas_leak		wrestler	wrestler		wrestler	wrestler
	Vrb	cause	hospitalise	think	think	hospitalise	think	think
	Obj	Fire	People	viral_infection	viral_infection	People	viral_infection	viral_infection
4	Sub	Bill	Voting	lineman	lineman	Piracy_in_Somalia	united_states_government	united_states_government
	Vrb	fail	worry	run	mind	release	unblock	change
	Obj	People	Official	president_of_united_states	group	Tanker	oil_tanker	soldier
5	Sub	Bus	Father	homo	male_person	Alarm	triggerable_device	triggerable_device
	Vrb	hit	end	be	be	be	stay	stay
	Obj	Cab	Searching	group	group	Silent		
6	Sub	Police						
	Vrb	found	kill	overwhelm	overwhelm	kill	overwhelm	overwhelm
	Obj	Body	People	group	group	People	group	group

Table 3: Part of the output results using the partial event similarity measure. (The entire output results are available at http://www.cems.uvm.edu/~bslee/eepat/results.pdf.)

			Predicted effect events from all cause-effect event pairs			Predicted effect events from the abstraction tree		
#	Resource	Input cause event	Exact single match	Exclusion set based abstracted multiple match	Ranked inclusion set based abstracted multiple match	Exact single match	Exclusion set based abstracted multiple match	Ranked inclusion set based abstracted multiple match
1	Sub	Storm	Marshal	objection	objection	People	Citizenry	group
	Vrb	cause	worry	learn	rule	be	stay	stay
	Obj	Fire		connection	support	Death	end	organic_phenomenon
2	Sub	Police_officer	Crowd	group_of_people	need			
	Vrb	kill	scramble	recognize	recognize	arrest	catch	check
	Obj	Gun		dependant	male_person		citizenry	group
3	Sub	Gas_leak		objection	objection			
	Vrb	cause	hospitalise	hospitalize	change	hospitalise	hospitalize	hospitalize
	Obj	Fire	People	citizenry	group	people	citizenry	group
4	Sub	Bill	Hope	president_lyndon_johnson	musical_artist	Hope	president_lyndon_johnson	musical_artist
	Vrb	fail	stir	get_down	move	stir	get_down	move
	Obj	People		people	magazine		people	magazine
5	Sub	Bus	Man	father_god	male_person			
	Vrb	hit	die	hurt	be	kill	die	overwhelm
	Obj	Cab		citizenry	group	People	citizenry	group
6	Sub	Police		penis	associate			
	Vrb	found	solve	ache	change	arrest	catch	check
	Obj	Body	Mystery	officer	person	People	Citizenry	group

Table 4: Part of the output results using the total event similarity measure. (The entire output results are available at http://www.cems.uvm.edu/~bslee/eepat/results.pdf.)

Dimension variable	Ratio	Mean of the ratio and the margin of error (confidence interval 95%)
Event similarity measure	Total event similarity / Partial event similarity	1.76 ± 0.13
Use of the abstraction tree	Using the abstraction tree / Not using the abstraction tree	1.16 ± 0.05
Prediction method	Exclusion set / Exact single match	0.84 ± 0.07
	Ranked inclusion set / Exact single match	0.84 ± 0.08
	Ranked inclusion set / Exclusion set	1.04 ± 0.04

Table 5: Summary of the user survey results.

zenry run' from ES and 'group run' from RIS. In a few cases, words used were so general that it was hard to interpret their meanings and the sentences constructed were not in a readable form – for example, 'allotment constitute script'. (Here, 'script' refers to dialogue, according to ConceptNet.)

The *baseline* EMS prediction showed the highest score, as expected. The reason was that the returned effect event was generally more readable since it was an *actual* event that occurred in the past. EMS, however, does not create a new effect event but returns the effect event that is paired with the best matching cause event.

Compared with EMS, both ES and RIS resulted in scores that are 16% lower. Given their abstraction mechanism, ES and RIS are expected to perform well when there are a large number of events that are semantically (or topically) similar to one another. In the current dataset, however, the number of events was small (i.e., about 5000) and the topics were sparsely distributed (i.e., many topics covering a small number of events) and as a result ES and RIS underperformed EMS. It was encouraging, however, that the performance drop was only 16%.

7 Conclusion

The project successfully implemented semantic web data to predict effect events. It proved the capability of the semantic web data to act as a knowledge base by enabling the construction of an abstraction tree. The knowledge that was transferred into this abstraction tree was used for prediction. This project showed how general semantic web datasets could be used to create a special purpose knowledge base.

The project is in its proof-of-concept prototype form. The immediate further work includes adding more semantic web dataset and adding more CE event pairs to empower the abstraction engine. It will results in much enhanced quality of the predicted effect event. Refining event semantics by adding such resources as time and location and extending effect event prediction by looking farther to find common resources will also result in enhanced prediction quality, albeit at a higher computational cost.

Acknowledgment

The authors thank Dr. Sèbastien Harispe and other anonymous reviewers for their invaluable comments on the original manuscript. The material presented in this paper is based upon work supported by the National Science Foundation under Grant No. IIS-1447634.

References

Sina Dami, Ahmad Abdollahzadeh Barforoush, and Hossein Shirazi. 2016. News events prediction using markov logic networks. *Journal of Information Science* .

DBpedia Association. 2017. DBpedia: Towards a public data infrastructure for a large, multilingual, semantic knowledge graph. http://wiki.dbpedia.org/.

Linmei Hu, Juanzi Li, Liqiang Nie, Xiao-Li Li, and Chao Shao. 2017. What happens next? Future subevent prediction using contextual hierarchical lstm. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*. pages 3450–3456.

Luminoso. 2017. ConceptNet: An open, multilingual knowledge graph. http://conceptnet.io/.

Max Planck Informatik Institut. 2017. YAGO: A high-quality knowledge base. http://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/yago/.

Kira Radinsky and Sagie Davidovich. 2012. Learning to predict from textual data. *Journal of Artificial Intelligence Research* 45(1):641–684.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2011. Learning causality from textual data. In *Proceedings of Learning by Reading for Intelligent Question Answering Conference*.

Kira Radinsky, Sagie Davidovich, and Shaul Markovitch. 2012. Learning causality for news events prediction. In *Proceedings of the 21st International Conference on World Wide Web*. ACM, pages 909–918.

Kira Radinsky and Eric Horvitz. 2013. Mining the web to predict future events. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*. pages 255–264.

SAPE Research Group. 2015. Hac - a Java class library for hierarchical agglomerative clustering. http://sape.inf.usi.ch/hac.

Wikipedia. 2017. Semantic similarity. https://en.wikipedia.org/wiki/Semantic_similarity.