9/10

2012 - 10 pages - 2021.12.~~03~~ 05

• path-based similarity distance - relatedness

Learning Causality for News Events Prediction

Kira Radinsky
CS Department
Technion–Israel Institute of Technology
Haifa, Israel
kirar@cs.technion.ac.il

Sagie Davidovich
CS Department
Technion–Israel Institute of Technology
Haifa, Israel
mesagie@gmail.com

Shaul Markovitch
CS Department
Technion–Israel Institute of Technology
Haifa, Israel
shaulm@cs.technion.ac.il

ABSTRACT

The problem we tackle in this work is, given a present news event, to generate a plausible future event that can be caused by the given event. We present a new methodology for modeling and predicting such future news events using machine learning and data mining techniques. Our Pundit algorithm generalizes examples of causality pairs to infer a causality predictor. To obtain precise labeled causality examples, we mine 150 years of news articles, and apply semantic natural language modeling techniques to titles containing certain predefined causality patterns. For generalization, the model uses a vast amount of world knowledge ontologies mined from LinkedData, containing 200 datasets with approximately 20 billion relations. Empirical evaluation on real news articles shows that our Pundit algorithm reaches a human-level performance.

Really

?←

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Knowledge acquisition; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems

General Terms

Algorithms, Experimentation

Keywords

News prediction, Future prediction, Web knowledge for future prediction

1. INTRODUCTION

When an agent, situated in a complex environment, plans its actions, it reasons about future changes to the environment. Some of the changes are a result of its own actions, but many others are a result of various chains of events not necessarily related to the agent. The process of observing an event, and reasoning about future events potentially *caused* by it, is called *causal reasoning.*

In the past, such complex environments were not accessible to computerized agents due to the limitation of their perceptive capabilities. The proliferation of the World Wide Web, however, changed all these. An intelligent agent can act in the virtual world of the Web, perceiving the state of the world through extensive sources of textual information, including Web pages, tweets, news reports, and online encyclopedias, and performing various tasks such as searching, organizing and generating information. To act intelligently in such a complex virtual environment, an agent must be able to perceive the current state, and reason about future states through causal reasoning. Such reasoning ability can be extremely helpful in conducting complex tasks such as identifying political unrest, detecting and tracking social trends, and generally support decision making by politicians, business people, and individual users.

While many works have been devoted to extracting information from text [3, 6], little has been done in the area of causality extraction [18, 19, 13, 12]. Furthermore, the algorithms developed for causality extraction are aimed at detection of causality pairs and cannot be used for causality prediction, i.e., given an event, generating new events that it can cause. The goal of the research presented in this paper is to provide algorithms that perform causal reasoning in textually represented unrestricted environments.

Our Pundit system, given an event represented in natural language, predicts future events it can potentially cause. To achieve this, we train the system with examples of causality relations automatically extracted from the Web. We first collect news reports from the last 150 years. We then use *textual causality patterns* (such as "X because Y", "X causes Y", etc.) to identify pairs of structured events that are supposedly related by causality (we do not rely on time correlation). The result is a semantically-structured causality graph of 300 million fact nodes connected by more than one billion edges. Our learning algorithm then uses large ontologies to generalize over the causality pairs and to predict causality of unseen cases. To evaluate our methods we test it on a news archive from 2010, which was not used during training. The results are judged by human evaluators.

To gain some intuition about the type of predictions the algorithm issues, we present here two examples. The algorithm, given the event "Magnitude 6.5 earthquake rocks the Solomon Islands", predicted that "Tsunami-warning will be issued in the Pacific Ocean". It learnt this based on past examples it was trained on, one of which was "Tsunami warning issued for Indian Ocean" after "7.6 earthquake strikes island near India". The predictive template inferred by Pundit was: if an earthquake occurs next to an Island, a tsunami warning will be issued for its nearest ocean. An additional example of a prediction issued by Pundit, is given the event "Cocaine found at Kennedy Space Center", it outputted the following predictions: "few people will be arrested", as the

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

past event "police found cocaine in lab" caused the event "2 people arrested". Such predictions provide evidence for the system ability to reason under changing conditions and issue logical predictions.

The contributions of this work are threefold: First, we present novel and scalable algorithms for generalizing causality pairs to causality rules. Second, we provide a new method for using casualty rules for predicting new events. Finally, we perform an empirical study on realistic problems judged by human raters. We make the extracted causality information publicly available for further research in the field[1].

2. RELATED WORK

We are not aware of any work that attempts to perform the task we face: receive arbitrary news event in natural language representation and predict events that it can cause. There are, however, several works that deal with related tasks. In computational linguistics, many studies deal with *extraction* of causality relations from text using causality patterns. These patterns are either manually crafted [18, 19, 13, 12], or automatically generated using machine learning techniques [5, 36, 32, 33, 1, 24, 9]. In textual entailment [14], the task is to identify texts that logically follow one another.

Another related task in natural language processing is the task of temporal information extraction – predicting the temporal order of events or time expressions described in text. Most of those approaches [25, 27, 21, 8, 39, 41] learn classifiers that predict a temporal order of a pair of events based on predefined pair's features. Shahaf and Guestrin [35] essentially detect a whole causality chain – given two news articles, they provide a coherent small number of news items connecting them. Similarly, some works in topic tracking [2], try to identify coherent story lines from news. More recently, some works [16, 16, 29] have explored the usage of large text mining techniques, applied on temporal corpus, such as news and books. These works focus mainly on how the culture develops, and what people's expectations and memories are from those times.

The above works, along with the works that deal with other entity and relation extraction [6] and common-sense knowledge acquisition from text [3, 34], tackle a task that is different from ours. Their task involves *identification* and *extraction* of causality relations, while ours involve *prediction* of events caused by a given event. Thus, our algorithm can generate events that were not observed before. Moreover, in Section 5.2 we describe a module of our algorithm that obtains training examples based on methods similar to the above.

We do not claim to create more precise information extraction methods, but rather how to leverage all this knowledge to perform an important AI task – future event prediction. We present novel methods of combining world knowledge with event extraction methods to represent coherent events. We then present novel methods for rule extraction and generalization using this knowledge.

3. EVENT REPRESENTATION

check it out!

The basic element of our algorithm is an event. We leverage elements from the work of Kim [20] and to design a representation scheme that allows us comparing events, generalizing them and reasoning about them. Given a set of (physical or abstract) objects O, Kim defined an event as a triplet $[O_i, P, t]$ where $O_i \subseteq O$ is a set of objects, P is a relation (or a property) over objects and t is a time interval. We propose a representation that further structures the event to have roles in addition to the property relation. Each event will be composed of a temporal action or state that the event's objects exhibit (P), one or more *actors* O_1 that conducted the action, one or more *objects* O_2 on which the action was performed, one or more *instruments* O_3 the action was conducted with, and one or more *locations* O_4 of the event. Formally, it is represented as an ordered set $e = \langle P, O_1, \ldots, O_4, t\rangle$, where P is the action, $O_i \subseteq O$ and t is a time-stamp. For example, the event "The U.S army destroyed a warehouse in Iraq with explosives", which occurred on October 2004, is modeled as: Destroy (Action); U.S Army (Actor); warehouse (Object); explosives (Instrument); Iraq (Location); October 2004 (Time). Illustration of more such events can be see in Figure 5.

The approaches in the literature for event representation can be divided to two groups: The first approach describes an event in the sentence level by an intact text or individual terms [5, 36]. However, representing the events "U.S army bombs a warehouse in Iraq", "Iraq attacks the U.S base" and "Terrorist base was attacked by the U.S marines in Kabul" using terms alone might yield that the first two events more similar than the first and last – as it lacks the understanding the actor of both events is a military group and that Kabul and Iraq are the locations of the events. The second approach, describes events in a syntax-driven way, where each element is mapped to a noun phrase [12, 19, 13, 9]. In our example, this representation again will not find the appropriate similarity of the events, as in the syntax level both the second and third event are similar.

Using these representations, it is hard to perform generalizations and comparison of events in a semantic or pragmatic way that takes into account all the elements composing an event. Our approach is semantic — it identifies actors, objects etc. This resembles the complex event representations presented in large knowledge ontolgies, such as Cyc [22]. This mapping of the atomic elements of each event (e.g., U.S Army) to semantic concepts (e.g., *actor*) provides a fertile ground for canonic representation of events that are both comparable and generalizable.

4. THE PREDICTION ALGORITHM

In this section we present Pundit – a learning algorithm that outputs a predictor g, which, given a present event, can predict its possible effects. During training, the learning algorithm generalizes over the given examples and produces an abstraction tree (AT) (Section 4.3). For each node in the AT, a prediction rule is generated based on the examples in the node (Section 4.4). In the prediction phase, the new event is matched to a node in the AT, and the associated rule is applied on it to produce the effect event.

4.1 Problem Definition

Let Ev be the universe of all possible events. We define a prediction function $f : 2^{Ev} \rightarrow 2^{Ev}$, that maps from a set of events to a set of future events (i.e., $f(e).t > e.t$). In this work, we focus on a subclass of this problem – functions of the form $g : Ev \rightarrow Ev$. Our solution is based on learning

[1] http://www.technion.ac.il/ kirar/Datasets.html

this function from examples. Assume there is a causality function g unknown to us. Assume we are given a set of examples $E = \{\langle e_1, g(e_1)\rangle, \ldots, \langle e_n, g(e_n)\rangle\}$, our goal is to produce a hypothesis $\hat{g}$ which is a good approximation of g.

4.2 Generalizing Over Objects and Actions

Our goal is to develop a learning algorithm that automatically induces a causality function based on examples of causality pairs. The inferred casualty function should be able to make its prediction for the given event, even if it was never observed before. For example, given the training examples ⟨earthquake in Turkey, destruction⟩ and ⟨earthquake in Australia, destruction⟩, and a current new event of "earthquake in Japan", a reasonable prediction would be "destruction". To be able to handle such predictions, we must endow our learning algorithm with generalization capacity. For example, in the above scenario, the algorithm needs the ability to generalize Australia and Turkey to countries, and to infer that earthquakes in countries might cause destruction. This type of inference and the knowledge that Japan is also a country enables the algorithm to predict the effects of new events based on patterns in the past.

To generalize over a set of examples, each consisting of a pair of events, we perform generalization over the components of these events. There are two types of such components – objects and actions.

To generalize over objects, we assume the availability of a semantic network $G_o = (V, E)$, where the nodes $V = O$ are the objects in our universe, and the labels on the edges are relations such as isA, partOf and CapitalOf. In this work, we consider one of the largest semantic network available, the LinkedData ontology [4], which we describe in detail in Section 5.

We define two objects to be similar if they relate to a third object in the same way. This relation can be a label or a sequence of labels in the semantic network. For example, Paris and London will be considered similar because both their nodes connect by the path $\xrightarrow{Capital-of}\xrightarrow{In-continent}$ to the node Europe. More formally:

Definition 1. *Let $a, b \in V$. A sequence of labels $L = l_1, \ldots, l_k$ is a **generalization path** of a, b, denoted by GenPath(a,b), if there exist two paths in G, $(a, v_1, l_1), \ldots (v_k, v_{k+1}, l_k)$ and $(b, w_1, l_1), \ldots (w_k, w_{k+1}, l_k)$, s.t. $v_{k+1} = w_{k+1}$.*

During generalization of the events over-generalization should be avoided – e.g., given two similar events, one occurring in Paris and one in London we wish to produce the generalization "city in Europe" ($\xrightarrow{Capital-of}\xrightarrow{In-continent} Europe$) rather than the most abstract generalization "city in a continent" ($\xrightarrow{Capital-of}\xrightarrow{In-continent}\xrightarrow{IsA} Continent$). That is, the minimal generalization of the objects.

Definition 2. *The **minimal generalization path**, denoted by $MGenPath(a, b)$, is defined as the shortest generalization path. We denote $dist_{Gen}(a, b)$ as the length of the $MGenPath(a, b)$.*

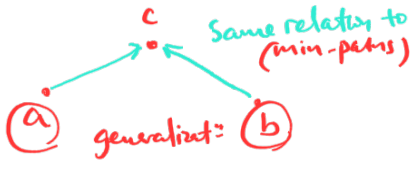
Path-based semantic distances, as the one above, were shown to be successful in many NLP applications. For example, a similiar distance was used to measure semantic relatedness of two words as a function of the length of the shortest path in a taxonomy (such as Wordnet or Wikipedia) connecting the two nodes representing the words [31, 37]. We build on this metric, and expand it to handle events, that are structured and can contain several objects from different ontologies.

```
Procedure MINIMAL GENERALIZATION PATH($G$)
  (1) $Q \leftarrow$ new Queue
  (2) Foreach $\{(a, c, l), (b, c, l) \in E(G)|$
                $a, b, c \in V(AT), l \in Lables\}$
    (2.1) $Mgen(a, b).Gen = c$
    (2.2) $Mgen(a, b).Pred = l$
    (2.3) $Mgen(a, b).Expanded = false$
    (2.4) $Q.enqueu((a, b))$
  (3) While $Q \neq \emptyset$:
    (3.1) $(a, b) \leftarrow Q.dequeu()$
    (3.2) If $Mgen(a, b).Expanded \neq true$:
      $Mgen(a, b).Expanded = true$
      (3.4) Foreach $\{(x, a, l), (y, b, l) \in E(AT)|$
                    $x, y \in V(AT), l \in Lables\}$
        (3.4.1) $Mgen(x, y).Gen = Mgen(a, b).Gen$
        (3.4.2) $Mgen(x, y).Pred = Mgen(a, b).Pred||l$
        (3.4.3) $Mgen(x, y).Expanded = false$
        (3.4.4) $Q.enqueu((x, y))$
  (4)Return $Mgen$
```

Figure 1: Procedure for calculating the minimal Generalization path for all object pairs

To efficiently produce $MGenPath$, we designed an algorithm (described in Figure 1), based on dynamic programming, that computes the $MGenPath$ for all object pairs in G. At stage 1 a queue that will hold all nodes with common generalization is initialized. At stage 2, the algorithm identifies all nodes (a, b) that have a common node (c) connecting to them via the same type of edge (l). c can be thought of as a generalization of a and b. The $Mgen$ structure maps a pair of nodes to their generalization ($Mgen.Gen$) and their generalization path ($MGen.Pred$). At stage 3, in a dynamic programming manner, the algorithm iterates over all nodes (a, b) in $Mgen$, for which we found a minimal generalization in previous iterations, and finds two nodes – one (x) connecting to a and one (y) connecting to b via the same type of edge l (stage 3.4). Thus, the minimal generalization of x and y is the minimal generalization of a and b, and the path is the $MGenPath$ of a, b with the addition of the edge type l. This update is performed in stages 3.4.1–3.4.4. Eventually, when no more nodes with minimal generalization can be expanded (i.e., cannot find two nodes that connect to them via the same edge type), the algorithm stops and return $Mgen$ (stage 4).

We also define a distance between actions using an ontology G_p, similarly to the way we defined distance between objects. Specifically, we use the VerbNet [17] ontology, which is one of the largest English verb lexicon. It has mapping to many other online resources, such as Wordnet [30]. The ontology is hierarchical and is based on a classification of the verbs to the Levin classes [15]. Using this ontology we describe the connections between verbs. Figure 6 shows a node in this ontology that generalizes the actions "hit" and "kick".



4.3 Generalizing Events

In order to provide significant support for generalization, we wish to find similar events that can be generalized to a single abstract event. In our example, we wish to infer that both ⟨earthquake in Turkey, destruction⟩ and ⟨earthquake in Australia, destruction⟩ are examples of the same group of events. Therefore, we wish to cluster the events in such a way that events, that have both similar causes and similar effects, will be clustered together. As in all clustering methods, a distance measure between the classification objects (in our case, the events) should be defined.

Let $e_i = \langle P^i, O^i_1, \ldots, O^i_4, t^i\rangle$ and $e_j = \langle P^j, O^j_1, \ldots, O^j_4, t^j\rangle$ be two events. In the previous subsection we defined a distance function between objects (and between actions). Here, we define the similarity of two events e_i and e_j to be the sum of distances between their objects and actions:

$$SIM(e_i, e_j) = dist^{G_p}_{Gen}(P^i, P^j) + \sum_{k=1}^{4} dist^{G_o}_{Gen}(O^i_k, O^j_k) \quad (1)$$

where, $dist^{G}_{Gen}$ is the distance function $dist_{Gen}$ in the graph G.

Likewise, a similarity between two pairs of cause-effect events $\langle c_i, e_i\rangle$ and $\langle c_j, e_j\rangle$ is defined as:

$$SIM(\langle c_i, e_i\rangle, \langle c_j, e_j\rangle) = SIM(c_i, c_j) + SIM(e_i, e_j) \quad (2)$$

Using the similarity measurement suggested above, the clustering process can be thought of as a grouping of the training examples in such a way that there is a low variance in their effects (similar to information gain methods where examples are clustered by their class) and have a high similarity in their cause. We use the HAC hierarchical clustering algorithm [11] as our clustering method. The algorithm starts by joining the closest event pairs together into a cluster, and keeps repeating the process by joining the closest two clusters together until all elements are linked together into a hierarchical graph of events which we call abstraction tree (AT).

During the prediction phase, the input cause event will be matched to one of the created clusters. To allow this, we assign to each node in AT a representative cause event, that is the event closest to the centroid of the node's cause events.

4.4 Causality Prediction Rule Generation

The last phase of the learning is the creation of rules that will allow us, given a cause event, to generate the predicted event. As the input cause event is matched against the node centroid, naively we could have returned the effect event of the matched centroid. This, however, would not provide us with the desired result. Assume an event e_i=”Earthquake hits Haiti“ occurred today, which is matched to a node represented by the centroid: “Earthquake hits Turkey”, whose effect is ”Red Cross help sent to Ankara“. Obviously, predicting that Red Cross help will be sent to Ankara because of an earthquake in Haiti is not reasonable. We would like to be able to abstract the relation between the past cause and past effect and learn a predicate clause that connects them, e.g. for ”Earthquake hits [Country Name]“ yield ”Red Cross help sent to [Capital of Country]“. During prediction, such a clause will be applied to the present input event e_i producing its effect with regard to e_i. In our example, the logical predicate clause would be *CapitalOf*, as *CapitalOf(Turkey)*= Ankara. When applied on the current event e_i: *CapitalOf(Haiti)* = Port-au-Prince, the output will now be ”Red Cross help sent to Port-au-Prince”. Notice that the application of the clauses can only be applied on certain types of objects – in our case, countries. The clauses can be of any length, e.g., the pair ⟨“suspect arrested in Brooklyn”, “Bloomberg declares emergency”⟩ produces the clause *Mayor(BoroughOf(x))*, as Brooklyn is a borough of New York, whose mayor is Bloomberg.

```
Procedure FindPath($curEntity, goalEntity, depth$)
  If $curEntity = goalEntity$ Return $\emptyset$
  Else
    If $depth = 0$ Return $NULL$
    Else
      Foreach $relation \in outEdges(curEntity)$
        $solution \leftarrow$ FindPath($relation(curEntity)$,
                  $goalEntity, depth - 1$)
        If $solution \neq NULL$
          Foreach $existingSolution \in Solutions$ :
            Return $Solutions \bigcup$
                $(existingSolution||relation||solution)$
      Return $Solutions$

Procedure LearnPredicateClause($\langle P^c, O^c_1, \ldots, O^c_4, t^c\rangle$,
$\langle P^e, O^e_1, \ldots, O^e_4, t^e\rangle, depth$)
  Foreach $O^c_i \in O^c$, $O^e_j \in O^e, k \in \{1 \ldots depth\}$
    Return Learn predicate Clause Inner($O^c_i, O^e_j, k$)
```

Figure 2: Procedure for inferring paths between two events in the causality graph

We will now show how to learn such clauses for each node in the AT graph. Recall that the semantic network graph G_O is an edge-labeled graph, where each edge is a triplet $\langle v_1, v_2, l\rangle$, where l is a predicate (e.g. “CapitalOf”). The rule-learning procedure is divided to two main steps: First, we find an undirected path p_i of length at most k in G_O between any object of the cause event to any object of the effect event. Note that we do not necessarily look for paths between two objects in the same role. In the above example, we found a path between the location of the cause event (Brooklyn) to the actor of the effect event (Bloomberg). Second, we construct a clause using the labels of the path p_i as the predicates. We call this the *predicate projection* of size k, $pred = l_1, \ldots, l_k$ from an event e_i to an event e_j. During prediction, the projections will be applied to the new event $e = \langle P^i, O_1, \ldots, O_4, t\rangle$ by finding an undirected path in G_O from O_i with the sequence of labels of *pred*. As k is unknown, the algorithm, for each training example $\langle c_t, e_t\rangle$ in a node in AT, finds all possible predicate paths with increasing sizes of k from the objects of c_t to the objects of e_t in the G_O graph. Each such path is weighted by the number of times it occurred in the node. The full procedure of predicate generation is described in Figure 2. The function $LearnPredicateClause$ calls the inner function $FindPath$ for different k sizes and different objects from the given cause and effect events. $FindPath$ is a recursive function trying to find in a graph a path of length k between the two objects. It returns the labels of such path if found.

```
Procedure PROPAGATION($e = \langle P^i, O_1, \ldots, O_4, t\rangle$)
  (1) $Candidates \leftarrow \{\}$
  (2) $Q \leftarrow$ new Queue
  (3) $Q.enqueu(G.root)$
  (4) While $Q \neq \emptyset$:
    (4.1) $cur \leftarrow Q.dequeu()$
    (4.2) Foreach $edge \in cur.OutEdges$:
          If $SIM(e, edge.Source) > SIM(e, edge.Destination)$:
            $Candidates \leftarrow Candidates \bigcup$
                  $\{(edge.Source, SIM(e, edge.Source))\}$
          Else :
            $Q.enqueu(edge.Destination)$
  (5) Return $Candidates$
```

Figure 3: Procedure for locating candidates for prediction

4.5 Prediction

Given a trained model $\hat{g}$, it can be applied to a new event $e = \langle P^i, O_1, \ldots, O_4, t\rangle$ and to output its effect. The process is divided into two main steps – propagation of the event in the AT retrieving all similar nodes that match the new event, and then the application of the node rule on the event to produce the effect of the event.

Given a new event, Pundit traverses the AT starting from the root. For each node in the search frontier, the algorithm computes the similarity ($SIM(e_i, e_j)$) of the input event to the centroid of each of the children on this node, and expands those children with better similarity than their parent. At the end, the algorithm returns the set of all nodes in the search frontier, sorted by their similarity to the input event. Intuitively, we try finding the nodes which are the least general but still similar to the new event. The full algorithm is illustrated in Figure 3. The algorithm saves a set of possible matched results ($Candidates$), and a queue holding the search frontier (Q). In stage 4, the algorithm traverses the graph. In stage 4.2, for each edge, the algorithm tests whether the similarity of the new event e to the parent node ($edge.Source$) is higher than to the child node ($edge.Destination$). If the test succeeds, the parent node, with its similarity score, is added to the possible results. After all edges are exposed, the algorithm returns the possible results in stage 5. A visualization of the process can be seen in Figure 4.

For each node retrieved in the previous stage, the node predicate projection, $pred$, is applied to the new event $e = \langle P^i, O_1, \ldots, O_4, t\rangle$ by finding an undirected path in G_O from O_i with the labels of $pred$. This rule generates a possible effect event based on the retrieved node.

The projection results are all the objects in the vertex reached. Formally, $pred$ can be applied if $\exists V_0 : O \subseteq V_0, \exists V_1 \ldots V_k : (V_0, V_1, l_1), \ldots (V_{k-1}, V_k, l_k) \in Edges(G_O)$. The projection results are all the objects $o \in V_k$. The projections results of all the nodes, are weighted by the similarity of the target cause to the node $MGen$ (for tie breaking).

5. CAUSALITY MINING PROCESS: IMPLEMENTATION DETAILS

In the previous section we present a high-level algorithm that requires training examples T, knowledge about entities G_O, and events' action classes P. One of the main challenges of this work was to build a scalable system to obtain those requirements.

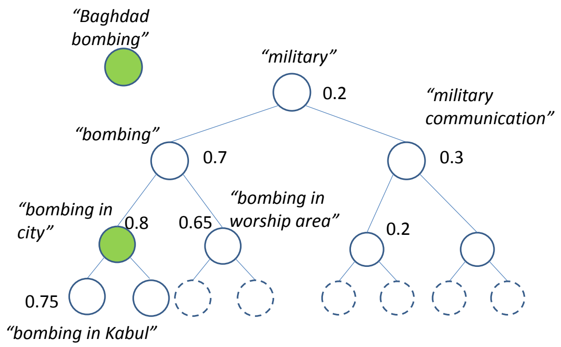


Figure 4: An event of a bombing in Baghdad is received as input. The system searches for the least general cause event it has observed in the past. In our case it is a generalized cluster: "bombing in city". The rule at this node now will be applied on Baghdad bombing to generate the prediction.

We present a system that mines news sources to extract events, constructs their canonical semantic model, and builds a causality graph on top of those events. The system crawled, for more than 4 months, several dynamic information sources. The largest information source was the New-York-Times archive, on which optical character recognition (OCR) was performed. The overall gathered data spans more than 150 consecutive years (1851 – 2009).

For generalization of the objects, the system automatically reads web content and extracts world knowledge. The knowledge was mined from structured and semi-structured publicly available information repository. The generation of the causality graph was distributed over 20 machines, using a Map-Reduce framework. This process efficiently unites different sources, extracts events, and disambiguates entities. The resulting causality graph is composed of over 300 million entity nodes, one billion static edges (connecting the different objects encountered in the events) and over 7 million causality edges (connecting events that were found by Pundit to cause each other). Each rule in the AT was generated based on an average of 3 instances with standard deviation of 2.

On top of the causality graph, a search and indexing infrastructure was built to enable search over millions of documents. This index allows a fast walk on the graph of events, enabling efficient inference capabilities during the reusing phase of the algorithm.

5.1 World Knowledge Mining

In this work we leverage the knowledge from several well-known ontologies to build the entity graph G_o. Our graph is composed of concepts from Wikipedia, ConceptNet [26], WordNet [30], Yago [38], and OpenCyc. The relations between the concepts (e.g., CapitalOf) are obtained from the LinkedData cloud project [4], where concepts are interlinked using human editors. The billion labeled edges of the graph

G_o are the predicates of those ontologies. Our system creates the entity graph by collecting the above content, processing feeds, and processing formatted data sets (e.g. Wikipedia). Our crawler then archives those documents in raw format, and transforms them into RDF format, using the interlinking information of LinkedData. We use SPARQL queries as a way of searching over the knowledge graph we create.

5.2 Causality Events Mining and Extraction

Our supervised learning algorithm requires many learning examples to be able to generalize well. As the amount of temporal data is extremely large, spanning over millions of articles, the goal of getting human annotated examples becomes impossible. We therefore provide an automatic procedure to extract labeled examples for learning causality from dynamic content. Specifically in this work, we used the New-York-Times archives for the years 1851 – 2009, WikiNews and BBC – over 14 million articles in total (see data statistics in Table 1). As we perform analysis on news titles alone, which are quite structured, the accuracy of this stage (performed on a representative subset of the data) is 78% (see Section 6.2.2). The system mines unstructured natural language text found in those titles, and searches for causal grammatical patterns. We construct those patterns using *causality connectors* as described in [40, 23]. Those connectors are divided to three groups:

1. Causal Connectives: in this set of connectors we used the words: because, as, and after as the connectors.
2. Causal prepositions: in this set of connectors we used the words: due to, and because of.
3. Periphrastic causative verbs: in this set of connectors we used the words: cause, and lead to.

We constructed a set of rules for extracting a causality pair. Each rule is structured as: ⟨Pattern, Constraint, Priority⟩, where Pattern is a regular expression containing a causality connector, Constraint is a syntactic constraint on the sentence on which the pattern can be applied, and Priority is the priority of the rule if several rules can be matched. For example, for the causality connector "after", the pattern "After [sentence1], [sentence2]" is used, with the constraint that [sentence1] cannot start with a number. This pattern can match the sentence "after Afghan vote, complaints of fraud surface" but will not match the sentence "after 10 years in lansing, state lawmaker Tom George returns". An additional pattern example is "[sentence1] as [sentence2]" with the constraint of [sentence2] having a verb. Using the constraint, the pattern can match the sentence "Nokia to cut jobs as it tries to catch up to rivals" is matched, but not the sentence "civil rights photographer unmasked as informer". The result of a rule application is a pair of sentences – one tagged as a cause, and one tagged as an effect.

Given a natural-language sentence (extracted form an article title), representing an event (either during learning or prediction), the following procedure transforms it into a structured event: lemma

1. Root forms of inflected words are extracted using a morphological analyzer derived from WordNet [30] stemmer. For example, in the article title from 10/02/2010: "U.S. attacks kill 17 militants in Pakistan", the words "attacks", "killed" and "militants" are transformed to "attack", "kill" and "militant" respectively.

Data Source	Number of Titles Extracted
New-York-Times	14,279,387
BBC	120,445
WikiNews	25,808

Table 1: Data Summary.

2. Part-Of-Speech tagging [28] is performed, and the verb is identified. The class of the verb is identified using the VerbNet vocabulary [17], e.g., kill belongs to P =murder class.
3. A syntactic template matching the verb is applied to extract the semantic relations and thus the roles of the words (see example in Figure 6). Those templates are based on VerbNet, which supplies for each verb class a set of syntactic templates. These templates match the syntax to the thematic roles of the entities in the sentence. We match the templates even if they are not continuous in the sentence tree. This allows the match of a sentence even where there is an auxiliary verb between the subject and the main transitive verb. In our example, the template is "NP1 V NP2" which transforms NP1 to "Agent", and NP2 to "Patient". Therefore, we match U.S. attacks to be the *Actor*, and the militant to be the *Patient* . If no template can be matched, the sentence is transformed into a typed-dependency graph of grammatical relations [28]. In the example, U.S. attacks is identified as the subject of the sentence (candidate for Actor), militants as the object (candidate for Patient), and Pakistan as the preposition (candidate for Location or Instrument, based on heuristics, e.g., locations lexicons). Using this analysis, we identify that the *Location* is Pakistan.
4. Each word in O_i is mapped to a Wikipedia-based concept. If a word matches more than one concept, we perform disambiguation by computing the cosine similarity between the body of the news article and the body of the Wikipedia article associated with the concept: e.g., U.S was matched to several concepts, such as: United States, University of Salford, and Us (Brother Ali album). The most similar by content was United States Wikipedia concept.
5. The time of the event t is the time of the publication of the article in the news, e.g., t =10/02/2010.

In our example, the final result is the event e = ⟨Murder-Class, United States Of America, Militant, NULL ,Pakistan, 10/02/2010⟩ . The final result of this stage is a causality graph composed of causality event pairs. Those events are structured as described in Section 3. We illustrate such a pair in Figure 5.

6. EMPIRICAL EVALUATION

A variety of experiments were conducted to test the performance and behavior of our algorithm.

6.1 Methodology

In this section we outline the methodology we used for our experiments. We provide two types of experiments – one evaluating the precision of the causality graph constructed,

This visualization is interesting

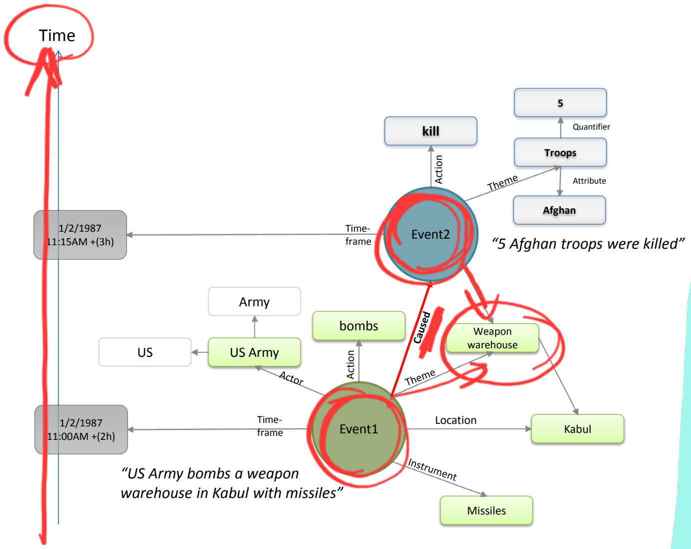


Figure 5: A pair of events in the causality graph. The first represents a cause event and the second represents the effect event. Both extracted from the title published on 1/2/1987: 5 Afghan troops killed after US army bombs warehouse in Kabul.

Class Hit-18.1		
Roles and Restrictions: Agent[int control] Patient[concrete] Instrument[concrete]		
Members: bang, bash, hit, kick, ...		
Frames:		
Example	Syntax	Semantics

Paula hit the ball	Agent V Patient	cause(Agent, E)
		manner(during(E), directedmotion, Agent)
		!contact(during(E), Agent, Patient)
		manner(end(E),forceful, Agent)
		contact(end(E), Agent, Patient)

Figure 6: VerbNet Template.

and the other evaluating the prediction accuracy of our system.

6.1.1 *Prediction Evaluation*

We implemented the algorithms described above and evaluated their performance. The prediction algorithm was trained using news articles from the period 1851 – 2009. The web resources snapshots mentioned in Section 5 dated until 2009. The evaluation is performed on separate data – Wikinews articles from the year 2010. We refer to this data as the *test data*. As in many supervised learning problems the evaluation is performed using human-tagged data. We performed two evaluation procedure – one comparing the system ability to predict (accuracy of prediction) and the plausibility of the predictions (to ensure that the predictions are not trivial and related to the cause event). The **plausibility evaluation** procedure is divided to the following steps:

1. Event identification – our algorithm assumes that the input to the predictor h is an event. To find news titles that represent an event, we randomly sample n headlines from the test data. For each headline a human is requested to decide whether the headline is an event which can cause other events. We denote the set of headlines labeled as event as E. We again randomly sample k titles from E. We denote this group as C.

2. Algorithm event prediction – on each event title $c_i \in C$, Pundit performs event extraction from the headline, and produces an event e_i^a with the highest score of being caused by the event represented by c_i. The result of this stage are the pairs: $\{(c_i, e_i^a)|c_i \in C\}$.

3. Human event prediction – a human is asked what the event $c_i \in C$ might cause. The instructions given to the human were to read the given event and to provide their best understanding on what the possible event it can cause. They were allowed to use any resource and were not limited by time. Human result is denoted as e_i^h. The human is requested to provide the answer in a structured manner (as our algorithm produces). The result of this stage are the pairs: $\{(c_i, e_i^h)|c_i \in C\}$.

4. Human evaluation of the results – Present m people with a triplet (c_i, e_i^h, e_i^a). We ask to evaluate the precision of the pair: (c_i, e_i^h) and the precision of (c_i, e_i^a), on a scale of 0-4 (0 is a highly impossible prediction and 4 is a highly possible prediction).

The **accuracy evaluation** is similar to the evaluation above, but in the third step we checked in the news (and other web resources) up to a year after the cause event, whether the predicted events appeared in the news. Raters were asked to supply confidence for their evaluations (as some events, like funerals, are not always reported in the news). We considered only confident and highly confident results in our evaluations.

All Human evaluation was conducted using Amazon Mechanical Turk (MTurk). We filtered the raters using a captcha and filtered out outliers. We performed the above mentioned experiments, with the values $n = 1500, k = 50, m = 10$.

6.1.2 *Extraction Evaluation*

As part of the analysis experiments of our algorithm, we provide evaluation of the information extraction techniques described in this work and used to train the system. Specifically, we provide two types of evaluation: event extraction evaluation and causality extraction evaluation. **Event Extraction evaluation** examines how well the event was extracted. Users were asked to evaluate on a scale of 1-5 how well the Action, Actor, Object, Instrument, and Time were extracted, given the original news title. We performed this evaluation on a randomly sampled 1000 news title, for each title we assigned 5 MTurkers. We filtered the raters using a captcha and filtered out outliers. Similarly, **Causality Extraction evaluation** evaluates the plausibility of the causality relation between two textual events. This evaluation indicates the precision of the causality templates we have crafted. MTurkers were shown two sentences the system believed had a cause and effect relation, and they were asked to evaluate the plausibility of this relation on a scale of 1-5. We performed evaluation over 500 randomly sampled pairs, for each we assigned 5 MTurkers.

	Highly Confident	Confident
Pundit	**58%**	**49%**
Humans	40%	38%

Table 2: Prediction accuracy for both human and algorithm predictions.

[0-1)	[1-2)	[2-3)	[3-4]	Average Ranking	Average Accuracy
0	2	19	29	3.08 ±0.19	**77%**
0	3	24	23	2.86 ±0.18	**72%**

Table 3: Plausibility results. The histogram of the rankings of the users for both human and algorithm predictions.

6.2 Results

In this section we provide the results for the two types of evaluation: the prediction evaluation (both plausibility and prediction evaluation) and the evaluation of the training data we provide as input to the system.

6.2.1 *Prediction Evaluation*

Accuracy evaluation results are reported in Table 2. Although the performance of Pundit was higher, a paired t-test on the k paired scores yielded a non statistically significant p-value (0.08). The results provide some evidence that the ability of the algorithm to predict future events is similar to that of the human ability to predict.

The **plausibility evaluation** yielded that Pundit's average prediction precision is 3.08/4 (3 is a "possible prediction"), and the human prediction average precision is $2.86 + -0.18/4$. For each event, we average the results of the m rankers, producing an averaged score for the algorithm performance on the event, and an averaged score for the human prediction (see Table 3). We performed a paired t-test on the k paired scores. The advantage of the algorithm over the humans was found to be statistically significant with $p < 0.05$. We can conclude now that the ability of the algorithm to produce plausible future events is better than the human ability to predict.

6.2.2 *Extraction Evaluation*

We have performed the experiments described in the methodology section to provide insights on the training data quality. The analysis of the quality indicated that the **averaged precision of the extracted causality** events is 3.11 out of 4 (**78%**), where 3 means a possible causality relation, and 4 means a highly possible causality relation. For comparison, other temporal rule extraction systems [8] reach precision of about 60%. For example, the casuality pair: "pulling over a car" → "2 New Jersey police officers shot", got a very high casuality precision score, as this is a plausible cause-effect relation. Although most causality extractions were with very high quality, the main reason for errors was that some events, though reported in the news and match the templates we have described, are reported as surprise to common sense casuality knowledge. For example, the following relation: "event the game heats up" → "the Yanks stay cool", was rated as a low casuality relation, and does not represent a plausible causality relation.

Action	Actor	Object	Instrument	Location	Time
93%	74%	76%	79%	79%	100%

Table 4: Event extraction precision based on the causality patterns.

Actor Matching	Object Matching	Instrument Matching	Location Matching
84%	83%	79%	89%

Table 5: Entity matching to ontology precision.

For every extracted causality event, we have calculated the precision of the semantic extraction of the event structure (Table 4). We conclude that the event extraction in this domain with the templates we have used has quite a high precision. For comparison, other works [7] for extracting entities for different types of relations reach 42−53% precision. The main reason for such high precision is the use of domain-specific templates for high precision (with lower recall). We performed additional experiments to evaluate the matching of every entity from the above to the world-knowledge ontology based on semantic similarity. The results as summarized in Table 5.

The **recall** of the entire process is **10%**. Our goal in general was to reach a high precision set of rules, from which generalization can be performed. We do not claim to have reached the highest performance with regard to causality extraction, but merely present a modular methodology that later studies can build upon.

We share the entire dataset of the extracted events online[2].

6.3 Discussion

We present in this section qualitative analysis of the results to have a better understanding of the algorithm strength and weaknesses. Given the event "Louisiana flood" the algorithm predicted that [number] people will flee. The prediction was based on the following past news articles: Residents of Florida flee storm and Hiltons; 150000 flee as hurricane nears north Carolina coast; a million flee as huge storm hits Texas coast; Thousands in Texas flee hurricane Ike; thousands flee as storm whips coast of Florida; at least 1000 flee flooding in Florida. The past events were generalized to the causality pair of "[Weather hazards] at [States of the Southern United States]" cause "[number] of people to flee". During the prediction, the event "Louisiana flood" was found most similar to the above generalized causality pair. The reusing classification function, outputted that [number] people will flee.

As another example, given the event "6.1 magnitude aftershock earthquake hits Haiti", it outputted the following predictions: "[number] people will be dead", "[number] people will be missing", "[number] magnitude aftershock earthquake will strike island near Haiti" and "earthquake will turn to United States Virgin Islands". While the first 3 predictions seem very reasonable, the fourth one is problematic. The rule the system learnt in this case is – natural disasters that hit countries next to a shore tend to affect near by countries. In our case it predicted that the earthquake will affect United States Virgin Islands, which are geographically

[2] http://www.technion.ac.il/ kirar/Datasets.html

Event	Human-predicted event	Pundit-predicted event
Al-Qaida demands hostage exchange	Al-Qaida exchanges hostage	A country will refuse the demand
Volcano erupts in Democratic Republic of Congo	Scientists in Republic of Congo investigate lava beds	Thousands of people flee from Congo
7.0 magnitude earthquake strikes Haitian coast	Tsunami in Haiti affects coast	Tsunami-warning is issued
2 Palestinians reportedly shot dead by Israeli troops	Israeli citizens protest against Palestinian leaders	War will be waged
Professor of Tehran University killed in bombing	Tehran students remember slain professor in memorial service	Professor funeral will be held
Alleged drug kingpin arrested in Mexico	Mafia kills people with guns in town	Kingpin will be sent to prison
UK bans Islamist group	Islamist group would adopt another name in the UK	Group will grow
China overtakes Germany as world's biggest exporter	German officials suspend tariffs	Wheat price will fall ?

huh?

Table 6: Human and algorithm predictions for events.

close to Haiti. However, the prediction "earthquake will turn to United States Virgin Islands" is not very realistic as an earthquake cannot change its course. It was created based on a match with a past example of a tornado hitting a country on a coast. The reason for that is the sparsity of the training. Both are natural disasters, and there were no negative examples or enough positive examples to support this distinction. However, we still find this example interesting, as it issues a prediction using spatial locality (United States Virgin Islands are [near] Haiti). Another example of the same problem is the prediction: ⟨ lightning kills 5 people, lightening will be arrested⟩, which was predicted based on training examples in which people who killed other people got arrested. More examples, out of the 50 in the test, can be seen in Table 6.

7. CONCLUSIONS

Many works have been done in information extraction and ontology building. In this work, we discuss how to leverage such knowledge into a large scale AI problem of events prediction. We present a system that is trained to predict events that might happen in the future, using a causing event as input. Each event is represented as a tuple of one predicate and 4 general semantic roles. The event pairs used for training were extracted automatically from news titles using simple syntactic patterns. Generalization to unseen events is achieved by: 1) creating an abstraction tree (AT) that contains entities from observed events together with their subsuming categories extracted from available online ontologies; 2) finding predicate paths connecting entities from causing events to entities in the caused events, where the paths are again extracted from available ontologies.

We discuss the many challenges of building such a system – starting from obtaining large enough dataset, the representation of the knowledge, and the inference algorithms needed for such a task. We perform *large-scale mining* and apply *natural language techniques* to transform the raw data of over 150 years of history archives into a structured representation of events, using a mined web-based object hierarchy and action classes. This shows the scalability of the proposed method, which is an important factor for any method that requires large amounts of data to work well. We also present that numerous resources, built by different people for different purposes (e.g. the different ontologies), can in fact be merged via a concept-graph to build a system that can work well in practice.

We perform *large-scale learning* over the large data corpus and present *novel inference* techniques. We consider both rule extraction and generalization. We propose novel methods for rule generalization using existing ontologies, which we believe can be useful for many other related tasks.

For future directions, we wish to investigate how to decay information about events in the system, as causality learnt in 1851 might be less relevant to the prediction in 2010. However, many common sense knowledge can still be used even if happened long time ago. Additional direction might include better event extraction, e.g., as proposed by Do et al. [10].

Our experimental evaluation showed that the predictions of the Pundit algorithm are at least as good as those of humans. We believe that our work is one of the first to harness the vast amount of information available on the web to perform prediction that is general purpose, knowledge based, and human like.

8. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *Proc. of JCDL*, pages 85–94, 2000.

[2] Amr Ahmed, Qirong Ho, Jacob Eisenstein, Eric P. Xing, Alexander J. Smola, and Choon Hui Teo. Unified analysis of streaming news. In *Proc. of WWW*, 2011.

[3] M. Banko and O. Etzioni. The tradeoffs between open and traditional relation extraction. In *Proc. of ACL-08: HLT*, 2008.

[4] C. Bizer, T. Heath, and T. Berners-Lee. Linked data – the story so far. *IJSWIS*, 2009.

[5] E. Blanco, N. Castell, and D. Moldovan. Causal Relation Extraction. In *Proc. of LREC*, 2008.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr., and T.M. Mitchell. Toward an architecture for never-ending language learning. In *Proc. of AAAI*, 2010.

[7] N. Chambers and D. Jurafsky. Template-Based Information Extraction without the Templates. In *Proc. of ACL*, 2011.

[8] N. Chambers, S Wang, and D. Jurafsky. Classifying temporal relations between events. In *Proc. of ACL (Poster)*, 2007.

[9] K. Chan and W. Lam. Extracting causation knowledge from natural language texts. *IJIS*, 20:327–358, 05.

[10] Q. Do, Y. Chan, and D. Roth. Minimally supervised event causality identification. In *EMNLP*, 2011.

[11] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *PNAS*, 95:14863–14868, 1998.

[12] D. Garcia. Coatis, an nlp system to locate expressions of actions connected by causality links. In *Proc. of EKAW*, 1997.

[13] R. Girju and D. Moldovan. Text mining for causal relations. In *Proc. of FLAIRS*, pages 360–364, 2002.

[14] O. Glickman, I. Dagan, and M. Koppel. A probabilistic classification approach for lexical textual entailment. In *Proc. of AAAI*, 2005.

[15] M. Palmer H. Dang, K. Kipper and J. Rosenzweig. Investigating regular sense extensions based on intersective levin classes. In *Proc. of Coling-ACL*, 1998.

[16] A. Jatowt and C.M Yeung. Extracting collective expectations about the future from large text collections. In *Proc. of CIKM*, 2011.

[17] N. Ryant K. Kipper, A. Korhonen and M. Palmer. Extending verbnet with novel verb classes. In *Proc. of LREC*, 2006.

[18] R. Kaplan and G. Berry-Rogghe. Knowledge-based acquisition of causal relationships in text. *Knowledge Acquisition*, 3:317–337, 1991.

[19] C. Khoo, S. Chan, and Y. Niu. Extracting causal knowledge from a medical database using graphical patterns. In *Proc. of ACL*, pages 336–343, 2000.

[20] J. Kim. Supervenience and mind. *Selected Philosophical Essays*, 1993.

[21] M. Lapata and A. Lascarides. Learning sentence-internal temporal relations. *JAIR*, 27:85–117, 2006.

[22] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.

[23] B. Levin and M. Rappaport Hovav. A preliminary analysis of causative verbs in english. *Lingua*, 92:35–77, 1994.

[24] D. Lin and P. Pantel. Dirt-discovery of inference rules from text. In *Proc. of KDD*, 2001.

[25] X. Ling and D. Weld. Temporal information extraction. In *Proc. of AAAI*, 2010.

[26] H. Liu and P. Singh. Conceptnet: A practical commonsense reasoning toolkit. *BT Technology Journal*, 22, 2004.

[27] I. Mani, B. Schiffman, and J. Zhang. Inferring temporal ordering of events in news. In *Proc. of HLT-NAACL 2003*, 2003.

[28] M. Marneffe, B. MacCartney, and C.D Manning. Generating typed dependency parses from phrase structure parses. In *Proc. of LREC*, 2006.

[29] J. Michel, Y.K Shen, A. Aiden, A. Veres, M. Gray, Google Books Team, J. Pickett, D. Hoiberg, D. Clancy, P. Norvig, J. Orwant, S. Pinker, M. Nowak, and E. Aiden. Quantitative analysis of culture using millions of digitized books. *Science*, 331:176–182, 2011.

[30] G. Miller. Wordnet: A lexical database for english. *CACM*, 38:39–41, 1995.

[31] H. Mili E. Bicknell Rada, R. and M. Blettner. Development and application of a metric to semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

[32] E. Riloff. Automatically Generating Extraction Patterns from Untagged Text. In *Proc. of AAAI*, 1996.

[33] E. Riloff and R. Jones. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proc. of AAAI*, 1999.

[34] L. Schubert. Can we derive general world knowledge from texts? In *Proc. of HLT 2002*, 2002.

[35] D. Shahaf and C. Guestrin. Connecting the dots between news articles. In *Proc. of KDD*, 2010.

[36] A. Sil, F. Huang, and A. Yates. Extracting action and event semantics from web text. In *Proc. of AAAI Fall Symposium on Commonsense Knowledge*, 2010.

[37] Michael Strube and Simone Paolo Ponzetto. Wikirelate! computing semantic relatedness using wikipedia. In *Proc. of AAAI*, 2006.

[38] F. Suchanek, G Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *Proc. of WWW*, 2007.

[39] M. Tatu and M. Srikanth. Experiments with reasoning for temporal relations between events. In *Proc. of COLING*, 2008.

[40] P. Wolff, G. Song, and D. Driscoll. Models of causation and causal verbs. In *Proc. of ACL*, 2002.

[41] K. Yoshikawa, S. Riedel, M. Asahara, and Y. Matsumoto. Jointly identifying temporal relations with markov logic. In *Proc. of ACL-IJCNLP*, 2009.