Learning Textual Graph Patterns to Detect Causal Event Relations

Bryan Rink, Cosmin Adrian Bejan, and **Sanda Harabagiu**
Human Language Technology Research Institute
The University of Texas at Dallas
Richardson, Texas 75080, USA
{bryan,ady,sanda}@hlt.utdallas.edu

Abstract

This paper presents a novel method for discovering causal relations between events encoded in text. In order to determine if two events from the same sentence are in a causal relation or not, we first build a graph representation of the sentence that encodes lexical, syntactic, and semantic information. In a second step, we automatically extract multiple graph patterns (or subgraphs) from such graph representations and sort them according to their relevance in determining the causality between two events from the same sentence. Finally, in order to decide if these events are causal or not, we train a binary classifier based on what graph patterns can be mapped to the graph representation associated with the two events. Our experimental results show that capturing the feature dependencies of causal event relations using a graph representation significantly outperforms an existing method that uses a flat representation of features.

1. Introduction

Automatic discovery of causal relations between textual events is a central task for various applications in Natural Language Processing (NLP) that require some form of *reasoning* such as probabilistic reasoning (Narayanan 1997), common sense reasoning (Mueller 2007), and question answering (Girju 2003).

An example of a causal relation between two events is expressed in the following sentence:

S1: *We* ***recognized*** *the problem and* ***took*** *care of it.*

This example encodes an ENABLEMENT relation, which is a special type of causal relation (in S1, the event *recognized* enables the event *took* to happen). Moreover, causal relations are closely related with temporal relations that hold between two events. For example, the ENABLEMENT relation corresponds to a BEFORE temporal relation (the event *took* happens only after the event *recognized* happened).

Most existing approaches for discovering causal relations train machine learning classifiers based on a flat representation of linguistic features. In this work, we propose a methodology that captures the contextual information of a pair of events by exploring various lexical, syntactic, and semantic features as well as the dependencies that exist between these features. For this purpose, we encode the features into a graph representation and cast the problem of causal relation discovery for a given pair of events from the same sentence into finding relevant graph patterns that capture the contextual information of the two events. This approach involves (i) how to determine which graph patterns are representative for expressing the causal information between the two events, and (ii) how to set up a machine learning framework that learns for each event pair which of these graph patterns decide if the events are causal or not.

The remaining part of this paper is organized as follows. Section 3 describes the methodology for building graph structures from text. Section 4 presents how graph patterns can be extracted from these graph representations. Section 5 describes how graph patterns map back to a graph representation corresponding to a given event pair. Section 6 describes the learning framework that decides whether an event is causal or not based on the the extracted graph patterns, Section 7 presents our experimental results, and finally, Section 8 discusses the most frequent errors made by our system.

2. Previous Work

Most approaches to causal relation detection have focused on a subset of the full set of causal relations in text. Although events can be expressed as verbs or as nouns in a nominalization, many existing systems only focus on one or the other. Work on nominal causal relation detection has been performed by Girju and Moldovan (2002) who detected verb expression patterns of the form "noun - verb - noun" and used constraints from WordNet to filter ambiguous occurrences. The SemEval 2007 Task 4 evaluation (Girju et al. 2007) contained nominal causal relations as a subset. The top system for CAUSE-EFFECT relations used an SVM classifier on several lexico-semantic features including the word stems and part of speech sequences between the nominals (Beamer et al. 2007). The work in (Khoo, Chan, and Niu 2000) used manually constructed dependency tree patterns to detect causal relations in medical domain documents. Their approach provides high precision extraction but requires manual effort to identify the patterns.

The problem of detecting causal relations between events has been studied by Bethard and Martin (2008). They consider the task of classifying whether, given two events that belong to the same sentence, one event can be considered a cause of the other. They train a classifier using surface features, WordNet hypernym and lexical files, as well as syntactic paths, and a scoring feature for the events based on web counts. We performed our evaluation on the same corpus used by Bethard and Martin (2008) to compare our structured feature approach against their flat feature approach.

3. Textual Graphs

Patterns capable of simultaneously matching text on the lexical, syntactic, and semantic level must utilize a representation that contains all of these types of information. Furthermore, there must be a mechanism for matching those patterns against an input text. To capture the syntactic and semantic relationships encoded in text, we represent both the text and the patterns in a graph representation as illustrated in Figure 1(a). The textual graphs are built up from nodes representing the tokens in a sentence, denoted *:tok:* in the graph. The token nodes have links to other nodes representing linguistic features or relationships of that token, such as the word, lemma, part-of-speech (POS), the root of its hypernym hierarchy, and syntactic parse tree leaf for that token. In addition, the two events from the sentence that are to be tested for a causal relationship, Event0 and Event1, have links from the *:tok:* node to an *Event* node which links to either *Event0* or *Event1*. There are also *next* edges between the POS nodes to enable capturing patterns that involve lexical dependencies.

Semantic relationships can be added to the graphs as well. For example, word sense disambiguation (WSD) can be integrated into the graphs by constructing a node for each synset needed for each token. As can be seen in Figure 1(c), each token node is then connected to the *highest* synset in that token's hypernymy chain. That node in turn is connected to the next highest synset and so forth, with the node for the exact sense detected coming last in the chain. This allows for patterns that match on tokens at a high hypernym level, or by including intermediate hypernyms, down to the most specific sense level. The event nodes are arranged in a similar general-to-specific structure for the same reason. We can also assign to event nodes their corresponding semantic roles. For instance, in Figure 1(a) we can add an edge representing the semantic role AGENT between the node representing the textual expression *We* and the event *recognized*.

Although this work describes the automatic detection of patterns, it is informative to examine the power of potential patterns. The text graph structure described above allows for patterns that act as conjunctions on the features of a token. For example, we could construct a pattern such as [*word:cause* → *:tok:* → *pos:VBP*] to recognize the verb form of the word *cause* rather than the noun form. This pattern fragment could then be joined with others to form an even more restrictive pattern. Looking at the patterns in this way (i.e., as conjunctions of feature restrictions) is reminiscent of rule learning such as that in (Cohen 1996). One difference is that the patterns we propose can capture textual dependencies (such as chains of POS) and also the structure of syntactic trees.

The structure we chose for the graphs is influenced by the patterns it enables, and also by the limitations of our pattern finding approach. For example, to allow for more generic and hence more powerful patterns, we could connect the *next* edges between *:tok:* nodes rather than *POS* nodes. However, increasing the powerfulness of the patterns in this way significantly increases the search space of possible patterns, beyond the memory capacity of the machine we ran the discovery algorithm on. Therefore connecting the POS nodes with *next* edges is a compromise that still allows some generality to be captured without generating too many candidate patterns.

4. Pattern Discovery

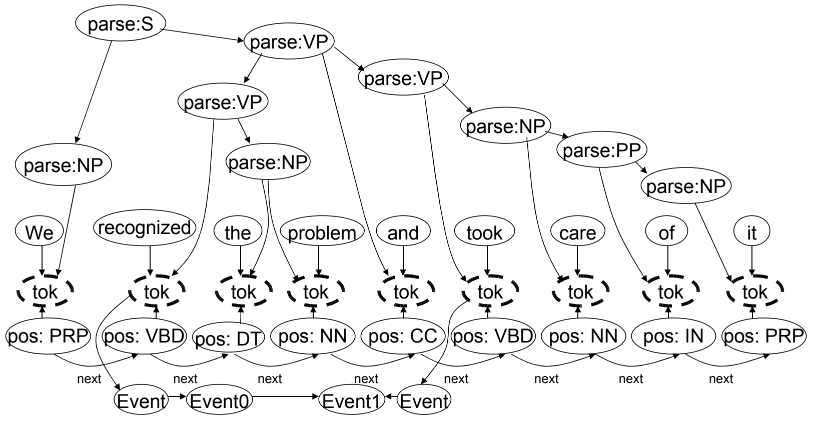
This section describes how graph patterns (or subgraphs) are extracted from a graph representation described in the previous section. Patterns are considered to match a target graph if all nodes and edges in the pattern correspond to equivalent nodes and edges in the target graph. Figure 1(b) shows a possible pattern that could be matched against the sentence in Figure 1(a). This pattern is in fact a subgraph of the graph representation associated with the example previously presented in S1. This pattern will match in sentences whose first event mention and the next token correspond to VBD and DT part-of-speeches respectively. The knowledge of whether or not this pattern matches a graph could be a useful indicator of whether the event might be causal. We explain in Section 6 how to use the presence of patterns within test sentences as features for a classifier.

Patterns of the form described above could be derived in several different ways, either manually constructed or automatically discovered. Many existing approaches, such as (Hearst 1998) and (Khoo, Chan, and Niu 2000) use manually constructed patterns. The textual graphs we describe can encompass the patterns from these existing approaches. For example, the use of sequences of words between named entities could be accomplished by adding a *next* link between the *word:* nodes in the graph, and by adding named entity nodes connected to the corresponding token nodes. We consider our approach a generalization of the works that automatically find patterns on sequences such as (Ravichandran and Hovy 2002). Rather than matching only sequences of words or paths in syntactic and dependency trees, we can find patterns on full textual graphs.

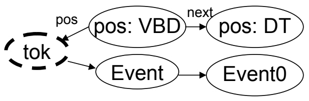
In recent years there has been much research on efficient detection of frequent subgraphs (Yan and Han 2002; Nijssen and Kok 2005; Kuramochi and Karypis 2001). While much of this research has focused on finding common substructures among chemical compounds, the algorithms can be used on arbitrary graphs, including the textual graphs described above. We use an implementation of the gSpan (Yan and Han 2002) frequent subgraph mining algorithm[1] to find all subgraphs which occur in at least 5% of the causal examples[2] in the training corpus. This balances the number

[1]`http://www2.informatik.uni-erlangen.de/EN/research/ParSeMiS/index.html`

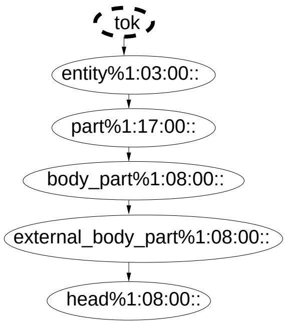
[2]Using both causal and non-causal examples provides no significant difference and increases run time.



(a) Graph example



(b) Basic pattern



(c) WSD graph example

Figure 1: An example text graph encoding syntactic information and a basic pattern, whose *:tok:* node matches the second *:tok:* node in the graph. (c) shows how WordNet hypernymy information can be added to the graph structure.

of patterns discovered against the exponential growth in the number of subgraphs. The CloseGraph (Yan and Han 2003) algorithm is used to obtain only maximal patterns, where no other pattern is a subgraph of any other pattern.

The gSpan algorithm finds all subgraphs which occur in a collection of graphs more frequently than a threshold called the *minimum support*. Since the total number of subgraphs is exponentially large, gSpan uses a pruning technique while searching the space of subgraphs. The algorithm builds increasingly larger subgraphs from smaller ones. The technique is based on the recognition that for any subgraph which occurs frequently in a collection of graphs, any larger graph containing that subgraph cannot occur more frequently. This leads to an approach that first selects all the subgraphs that have only one node and whose frequency is above the minimum support. In the second step, these one-node subgraphs are extended to 2-node subgraphs, while discarding extensions whose frequency is below the minimum support. This process continues iteratively until all the frequent subgraphs are discovered.

Therefore, our approach for discovering patterns for matching causal event relations consists of: (i) building graph representations for all the sentences that contain causal examples in the training set; and (ii) using gSpan to detect the frequent patterns.

5. Pattern Matching

Since the frequent patterns were built from the training set, we need to find a way for mapping those patterns onto the graph representations from the test set. This is, in fact, the graph homomorphism problem. Specifically, we need to test for graph monomorphism (an injective homomorphism) between the extracted patterns and the test graphs, where each node (and edge) in a pattern must map to a unique node (and edge) in the test graph and two nodes (edges) in the pattern cannot map to the same node (edge) in the test graph.

Although graph monomorphism is an NP-complete problem, for the size of the textual graphs considered this problem is computationally tractable. For pattern matching, we cast the problem of graph monomorphism as a constraint satisfaction problem and run a constraint satisfaction solver[3]to determine satisfiability. As an optimization, we employ a pre-filtering step to ensure that all the node and edge labels from patterns are present in the test graphs.

6. Graph Classification

The approach to graph classification we use is based on representing each sentence as a binary vector from the patterns that match the sentence. Using the above pattern matching technique, we are able to determine which patterns match which sentences. For each sentence in the corpus, we construct a binary vector in which the i^{th} element of the vector indicates whether the i^{th} pattern matches that sentence or not. These vectors then become the representation of each sentence that we pass to an SVM classifier[4], to detect causal relations. Similar to Bethard and Martin (2008), we optimize the F1 measure rather than accuracy by adjusting the cost factor, j, of the SVM classifier. Using all the discovered patterns generally leads to inferior results compared to ranking the patterns first and only using the top k to construct the binary vectors[5].

Therefore, the patterns that come out of gSpan are ranked according to their probability using the Fisher exact test of how likely that pattern's matches are given an assumption of independence from the causal relation. For a 2x2 table this probability is calculated as

$$p = \frac{(a+b)!(c+d)!(a+c)!(b+d)!}{n!a!b!c!d!}$$

[3]`http://bach.istc.kobe-u.ac.jp/cream/`
[4]SVM^{light} from `http://svmlight.joachims.org/`
[5]This effect is shown later in Figure 2

	Match	¬ Match
Causal	18	189
¬ Causal	5	485

Table 1: Contingency table example for a pattern.

where the first row is $<a, b>$, the second row is $<c, d>$, and $n{=}a+b+c+d$. Under this measure, lower scores are better because they indicate that the counts were not likely to have come from independent variables. Table 1 provides a contingency table example for one of the discovered patterns. As listed in the table, this pattern matches on causal sentences 18 times and non-causal sentences only 5 times. Considering that there are more than twice as many non-causal sentences than causal sentences in the training corpus, this pattern is a significant indicator of a causal relation.

Since gSpan finds all patterns, many of the top patterns are near-duplicates of each other. Patterns P1 and P2 provide a simple example of this phenomenon.

P1: *word:someone* → *:tok:* → *pos:NN*

P2: *word:someone* → *:tok:* → *lemma:someone*

These patterns match on the exact same sentences and therefore using both of them in a classifier would not be productive. There are similar pairs of patterns where both match on almost all of the same examples but with small differences. To avoid these duplicate patterns, we employ a sequential cover algorithm similar to the one described in (Thoma et al. 2009). This is outlined in Algorithm 1.

Algorithm 1 Positive/Negative Sequential Cover

```
Input: Set of patterns $P$ sorted by descending significance;
    positive and negative training examples $T_p$ and $T_n$, respectively
Output: Ranked set of patterns $R$
  $R = \emptyset$
  while $|T_p| \geq 0 \wedge |T_n| \geq 0 \wedge |P| > 0$ do
    $p$ = first pattern in $P$
    if $p$ is causal and matches $\geq 1$ graph in $T_p$ then
      $T_p = T_p \setminus \{g | g \in T_p$ is matched by $p\}$
      $R = R \cup \{p\}$
    else if $p$ is non-causal and matches $\geq 1$ graph in $T_n$ then
      $T_n = T_n \setminus \{g | g \in T_n$ is matched by $p\}$
      $R = R \cup \{p\}$
    end if
    $P = P \setminus \{p\}$
  end while
```

In Algorithm 1, we consider a pattern to be either causal or non-causal. Moreover, since approximately 30% of the training sentences are causal, we consider a pattern to be causal if more than 30% of its matches are causal. This algorithm will discard those patterns that only match graphs that have already been covered by at least one pattern previously ranked. To obtain a rank for the discarded patterns, Algorithm 1 is run again on those patterns until all patterns have been ranked. Once the patterns have been ranked, the top k are used to create the binary vector assigned to each sentence. Only these top k patterns are given to the classifier.

7. Experiments

We evaluated our system on a corpus annotated with event causal relations described in (Bethard et al. 2008). This corpus contains 1,000 sentences, where each sentence annotates two events in a causal or non-causal relation. The event pairs from each sentence are expressed as verbs in a conjunction since such occurrences are often temporally or causally related. The experimental setup of our binary classifier uses the same train/test split as used in (Bethard and Martin 2008), which contains 697 and 303 sentences for training and test splits respectively. To compare our system with the system proposed by Bethard and Martin (2008), in which the two major configurations of features correspond to a syntactic and a semantic feature set, we also use a syntactic and a semantic set of annotations in the graphs.

In our experiments, we build graph representations using the following set of features:

- Word: The surface word (e.g., *ran*);
- POS: The part of speech (e.g., *VBD*);
- Parse: A syntactic parse tree as in Figure 1(a);
- Stem: The WordNet stem (e.g., *run*);
- VerbOcean: Semantic links between verbs taken from VerbOcean (Chklovski and Pantel 2004), without weights;
- Dep (Dependency parse)[6]: Each dependency link is represented by a node in the graph, with a directed edge coming in to it from the source of the dependency, and a directed edge leaving the node to the *:tok:* node for the destination of the dependency. An example can be seen from the pattern in Figure 4;
- WSD: The hypernym chain for the senses from word sense disambiguation (Mihalcea and Csomai 2005);
- FNType: Using the UTD-SRL semantic parser (Bejan and Hathaway 2007), we extract for each event the semantic frame it evokes;
- Tmp: Manually annotated temporal links from the corpus.

Configuration	Patterns	k	j	P	R	F1
Bethard & Martin Syntactic	-	-	-	24	80	37.4
Bethard & Martin Semantic	-	-	-	27	64	38.1
Word+POS+Parse	16377	1000	1.5	26	78	38.9
Word+POS+Parse+Stem	16415	1000	1.5	27	70	39.1
Word+POS+Dep	14800	200	5.0	26	70	38.3
Word+POS+Dep+VerbOcean	14811	200	3.0	29	59	38.8
Word+POS+Dep+FNType	14984	200	3.0	31	63	41.7
Word+POS+Dep+WSD	39341	400	3.0	33	61	42.9
Bethard & Martin All+Tmp	-	-	-	47	59	52.4
Word+POS+Parse+Tmp	18464	10	1.5	51	66	57.5
Word+POS+Dep+WSD+Tmp	43275	10	1.5	52	66	57.9

Table 2: Comparison using different sets of annotations. k is the number of top patterns passed to the classifier, and j is the cost ratio used for the SVM. Patterns represents the total number of patterns discovered.

As listed in Table 2, the system proposed by Bethard and Martin (2008) achieves F-measures of 37.4 and 38.1 for the

[6]We use the Stanford dependency parser `http://nlp.stanford.edu/software/lex-parser.shtml`

k j	10	100	200	300	400	500	1000	2000	5000	∞
1	18.0	13.7	15.2	15.2	15.2	15.0	15.7	20.0	14.1	5.6
1.5	17.5	17.3	18.5	20.6	20.3	21.0	28.0	29.6	25.4	19.8
2	17.5	19.1	27.5	28.3	29.9	31.5	32.0	33.1	29.1	29.8
3	17.5	35.1	35.0	36.7	34.1	34.2	34.5	34.3	33.1	36.6
4	34.8	34.8	36.5	36.6	38.0	35.2	33.9	33.3	34.7	33.8
5	34.8	35.2	37.9	38.5	38.0	37.6	36.7	35.1	34.0	33.1
6	34.8	35.2	36.9	**38.9**	37.4	37.2	36.2	36.1	34.7	33.6
7	34.8	33.7	37.5	38.2	36.0	36.7	36.2	35.2	33.3	33.9
8	34.8	35.2	35.1	37.6	36.2	36.0	35.9	36.3	35.0	33.9
10	34.8	34.3	33.5	36.2	34.0	35.6	36.0	33.8	34.5	33.9
50	34.8	34.8	35.3	35.6	34.3	35.1	32.3	34.1	34.2	33.9
100k	34.8	34.8	35.3	35.6	34.3	35.1	32.3	34.1	34.2	33.9

Table 3: Values of F1 on the test set for various values j and k.

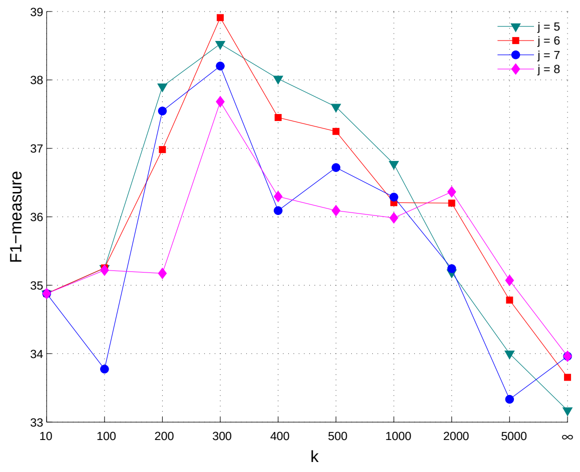


Figure 2: F1 scores corresponding to Table 3.

syntactic and semantic feature sets respectively. We compare these results against the results obtained by our system using various sets of features. All the experiments performed by our system use the automatic pattern discovery and classification process outlined above. The scores shown in Table 2 are the best scores achieved over a range of parameter choices. The two parameters are k, the number of top patterns to use, and the cost factor j used by the SVM to weight positive instances more. Training these parameters using cross validation on the training set proved to be difficult for several reasons. First, the distribution of causal events is different between the training and test sets. The training set consists of 29.7% causal relations, while only 21.5% of the relations in the test set are causal.

A further impediment is the amount of time required to discover the patterns, especially across many folds where the patterns have to be discovered for each fold. For these reasons, we present the score for the best choice of parameters in Table 2 and show the full range of parameter values and the scores obtained for those values on the Word+POS+Parse model in Table 3. Figure 2 shows a graphical representation of how the different parameter values affect the F1 score.

Table 2 compares the scores achieved for various combinations of syntactic and semantic annotations

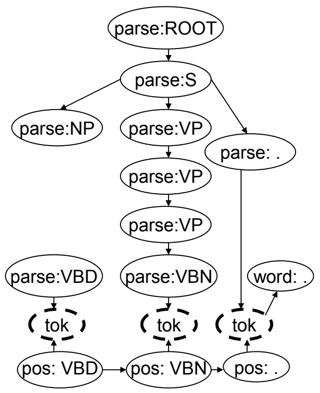


Figure 3: A highly ranked causal pattern that detects constructs such as passive voice.

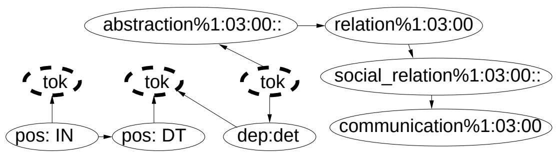


Figure 4: A semantic pattern detected by the system that looks for a phrase involving some form of communication.

in the graph. We used the Word+POS+Parse and Word+POS+Parse+Stem configurations as a comparison against the syntactic and semantic feature sets used in (Bethard and Martin 2008). These experiments indicate that using an appropriate configuration of parameters, the results achieved by our system outperform the results reported in (Bethard and Martin 2008). When using only a subset of their features in our graph representations, our system achieves a higher score. The best results were obtained by changing the parse tree with a dependency tree and using the word sense disambiguated hypernym chains, which achieved an F1 score more than 4 points higher than the existing system. The last three systems listed in Table 2 make additional use of a manually annotated temporal relation type between the two events. The higher scores prove that the temporal relations between causal events have a significant role in discovering causal relations.

Figures 3 and 4 show some of the top discovered patterns. These patterns are complex and detect structures in the text that would be difficult for a human to choose a priori. The pattern in Figure 3 shows a highly ranked pattern for the Word+POS+Parse graphs. This pattern matches causal examples 14 times in the training set, but doesn't match any non-causal sentences. The most salient characteristic of this pattern is that it will only match cases where a token with the part of speech VBN (past participle) immediately follows a VBD token (past tense). The rest of the pattern positions this occurrence within three levels of VP nesting in the parse tree. An example of a sentence from the training set matched by this pattern is given in S2:

S2: *Hells Angels was* ***formed*** *in 1948 and* ***incorporated*** *in 1966 .*

For this sentence, the pattern captures the passive voice verbal phrase, *"was formed"*, which is matched by VBD→VBN.

Another high ranking pattern, taken from the Word+POS+Dep+WSD graphs, is depicted in Figure 4. One sentence on which this pattern can match is:

S3: *Under the agreement, Westinghouse will be able to **purchase** smaller combustion turbines from its Japanese partner, and **package** and sell them with its own generators and other equipment.*

This is an example of an ENABLEMENT causal relation as described in Section 1.The pattern looks for a specific kind of phrase containing a preposition followed by a determiner followed by a word which is a hyponym of *communication*. In the example, this matches on the expression *"Under the agreement"*. Examples of other textual expressions this pattern matched in the training set are: *"[out] of the question"*, *"In that decision"*, *"In the affidavits"*, and *"that the offer"*. These phrases might be indications that the sentence is referring to a chain of events, which would imply a causal relation.

8. Error Analysis

A number of mistakes are made between statement events and the things being stated. For instance, the best run incorrectly marks this sentence as causal: *"Delmed **said** yesterday that Fresenius USA would begin distributing the product and that the company is **investigating** other possible distribution channels."* The investigation is not causally related to the fact that the company made a statement about it. A related sentence the system incorrectly marked as non-causal is: *"People have been **seeing** headline after headline after headline and **saying** : ..."* Here the event *saying* is caused by the event *seeing*. The order of the *saying* events in these two examples is different and is probably an important clue for causality detection. We also hypothesize that there simply weren't enough examples in the training set to learn this order clue for *saying* events.

Another area of errors which could be alleviated with additional data is cue phrases. As an example, the following sentence was marked non-causal: *"... they were not necessary to prove historical points , **failed** the fair-use test and therefore **infringed** copyright."* In this example, the cue phrase *therefore* provides evidence that the events are causally related. These cue phrases could either be learned with additional training data or a list of such phrases could be compiled and implemented as an additional annotation in the graph structure to reduce the learning burden and amount of training data needed.

9. Conclusion

This paper outlined a new approach for discovering causal relations between events in text using graph patterns as features to a classifier. This approach has the advantage that feature combinations and textual structure are automatically discovered rather than manually selected. This has the consequence that adding a new feature to the textual graphs could be equivalent to adding many flat features in a classical system, reducing the manual effort required to explore various combinations. In addition, using a graph representation to also capture the dependencies between features shows that this approach achieves better results when compared with a method that uses a flat representation on the same set of features.

References

Beamer, B.; Bhat, S.; Chee, B.; Fister, A.; Rozovskaya, A.; and Girju, R. 2007. UIUC: A knowledge-rich approach to identifying semantic relations between nominals. In *ACL SemEval07*.

Bejan, C. A., and Hathaway, C. 2007. UTD-SRL: a pipeline architecture for extracting frame semantic structures. In *ACL SemEval07*, 460–463.

Bethard, S., and Martin, J. H. 2008. Learning semantic links from a corpus of parallel temporal and causal relations. In *ACL*.

Bethard, S.; Corvey, W.; Klingenstein, S.; and Martin, J. H. 2008. Building a corpus of temporal-causal structure. In *LREC*.

Chklovski, T., and Pantel, P. 2004. Verbocean: Mining the web for fine-grained semantic verb relations. In *EMNLP*.

Cohen, W. 1996. Learning trees and rules with set-valued features. In *AAAI 1996*, 709—716.

Girju, R., and Moldovan, D. 2002. Mining answers for causation questions. In *Proc. The AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases*.

Girju, R.; Nakov, P.; Nastase, V.; Szpakowicz, S.; Turney, P.; and Yuret, D. 2007. SemEval-2007 task 04: classification of semantic relations between nominals. In *SemEval07*. ACL.

Girju, R. 2003. Automatic Detection of Causal Relations for Question Answering. In *ACL 2003 workshop on "Multilingual Summarization and Question Answering - Machine Learning and Beyond"*.

Hearst, M. A. 1998. Automated discovery of wordnet relations. *WordNet: An electronic lexical database* 131151.

Khoo, C. S. G.; Chan, S.; and Niu, Y. 2000. Extracting causal knowledge from a medical database using graphical patterns. In *ACL*, 336–343.

Kuramochi, M., and Karypis, G. 2001. Frequent subgraph discovery. In *IEEE International Conference on Data Mining*.

Mihalcea, R., and Csomai, A. 2005. SenseLearner: Word sense disambiguation for all words in unrestricted text. In *ACL 2005*.

Mueller, E. T. 2007. Modelling Space and Time in Narratives about Restaurants. *Literary and Linguistic Computing*.

Narayanan, S. 1997. *KARMA: Knowledge-based Action Representations for Metaphor and Aspect*. Ph.D. Dissertation, University of California, Berkeley.

Nijssen, S., and Kok, J. N. 2005. The gaston tool for frequent subgraph mining. *Electronic Notes in Theoretical Computer Science* 127(1):77–87.

Ravichandran, D., and Hovy, E. 2002. Learning surface text patterns for a question answering system. In *ACL*.

Thoma, M.; Cheng, H.; Gretton, A.; Han, J.; Kriegel, H. P.; Smola, A.; Philip, S. Y. L. S.; Yan, X.; and Borgwardt, K. 2009. Near-optimal supervised feature selection among frequent subgraphs. In *SIAM Intl Conf. on Data Mining*.

Yan, X., and Han, J. 2002. gSpan: graph-based substructure pattern mining. In *(ICDM'02)*.

Yan, X., and Han, J. 2003. CloseGraph: Mining closed frequent graph patterns. In *ACM SIGKDD*, 286–295.