Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Leveraging Dependency Regularization for Event Extraction

Kai Cao, Xiang Li, and **Ralph Grishman**
Computer Science Department
New York University
719 Broadway, New York, NY, 10003
{kcao, xiangli, grishman}@cs.nyu.edu

Abstract

Event Extraction (EE) is a challenging Information Extraction task which aims to discover event triggers with specific types and their arguments. Most recent research on Event Extraction relies on pattern-based or feature-based approaches, trained on annotated corpora, to recognize combinations of event triggers, arguments, and other contextual information. These combinations may each appear in a variety of linguistic forms. Not all of these event expressions will have appeared in the training data, thus adversely affecting EE performance. In this paper, we demonstrate the overall effectiveness of *Dependency Regularization* techniques to generalize the patterns extracted from the training data to boost EE performance. We present experimental results on the ACE 2005 corpus, showing improvement over the baseline system, and consider the impact of the individual regularization rules.

Introduction

Event Extraction (EE) involves identifying instances of specified types of events and the corresponding arguments in text, which is an important but difficult Information Extraction (IE) task. Associated with each event mention is a phrase, the event trigger (most often a single verb or nominalization), which evokes that event. More precisely, our task involves identifying event triggers associated with corresponding arguments and classifying them into specific event types. For instance, according to the ACE 2005 annotation guidelines[1], in the sentence "*[**She**] was **killed** by [**an automobile**] [**yesterday**]*", an event extraction system should be able to recognize the word "*killed*" as a trigger for the event DIE, and discover "*an automobile*" and "*yesterday*" as the *Agent* and *Time* Arguments. This task is quite challenging, as the same event might appear in the form of various trigger expressions and an expression might represent different events in different contexts.

Most recent research on Automatic Content Extraction (ACE) Event Extraction relies on pattern-based or feature-based approaches to building classifiers for event trigger and argument labeling. Although the training corpus is quite large (300,000 words), the test data will inevitably contain some event expressions that never occur in the training data. To address this problem, we propose several *Dependency Regularization* methods to help generalize the syntactic patterns extracted from the training data in order to boost EE performance. Among the syntactic representations, dependency relations serve as important features or part of a pattern-based framework in IE systems, and play a significant role in IE approaches. These proposed regularization rules will be applied either to the dependency parse outputs of the candidate sentences or to the patterns themselves to facilitate detecting the event instances. The experimental results demonstrate that our pattern-based system with the expanded patterns can achieve substantial improvement over the baseline, which is an advance over the state-of-the-art systems.

The paper is organized as follows: we first describe the role of dependency analysis in event extraction and how dependency regularization methods can improve EE performance. In the sections which follow, we describe our EE systems including the baseline and enhanced system utilizing dependency regularization, we present experimental results, and we discuss related work.

Dependency Regularization

The ACE 2005 Event Guidelines specify a set of 33 types of events, and these have been widely used for research on event extraction over the past decade.

Some trigger words are unambiguous indicators of particular types of events. For example, the word *murder* indicates an event of type DIE. However, most words have multiple senses and so may be associated with multiple types of events. Many of these cases can be disambiguated based on the semantic types of the trigger arguments:

- *fire* can be either an ATTACK event ("*fire a weapon*") or END-POSITION event ("*fire a person*"), with the cases distinguishable by the semantic type of the direct object. *discharge* has the same ambiguity and the same disambiguation rule.
- *leave* can be either a TRANSPORT event ("*he left the building*") or an END-POSITION event ("*he left the administration*"), again generally distinguishable by the type of the direct object.

[1]https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf

Given a training corpus annotated with triggers and event arguments we can assemble a set of frames and link them to particular event types. Each frame will record the event arguments and their syntactic (dependency) relation to the trigger. When decoding new text, we will parse it with a dependency parser, look for a matching frame, and tag the trigger candidate with the corresponding event type.

One complication is that the frames may be embedded in different syntactic structures: verbal and nominal forms, relative clauses, active and passive voice, etc. Because of the limited size of the training corpus, some triggers will appear with frames not seen in the training corpus. To fill these gaps, we will employ a set of *dependency regularization* rules which transform the syntactic structure of the input to reduce variation.

We describe here three of the regularization rules we use:

1. Verb Chain regularization
2. Passive Voice Regularization
3. Relative Clause Regularization

Verb Chain Regularization

We use a fast dependency parser (Tratz and Hovy 2011) that analyzes multi-word verb groups (with auxiliaries) into chains with the first word at the head of the chain. *Verb Chain (vch) Regularization* reverses the verb chains to place the main (final) verb at the top of the dependency parse tree. This reduces the variation in the dependency paths from trigger to arguments due to differences in tense, aspect, and modality. Here is an example sentence containing a verb chain:

Kobe has defeated Michael . (1)

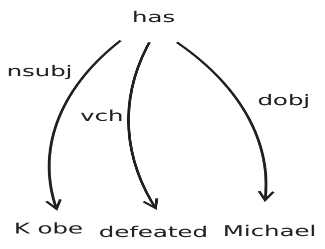

Figure 1: Original Dependency Tree

In the above sentence, "*has*" is originally recognized as the root of the dependency parse tree, while "*defeated*" is the dependent of the word "*has*". The dependency label of `(has, defeated)` is *vch*. However, the semantic head of the sequence (the word which determines the event type) is the last word in the verb chain. To bring the trigger and its arguments closer, we regularize the dependency structure by making the last verb in this chain the head of the whole verb chain. A further example:

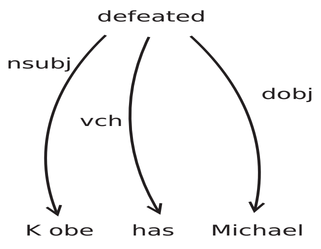

Figure 2: Dependency Tree with *Verb Chain Regularization*

You must come to school tomorrow . (2)

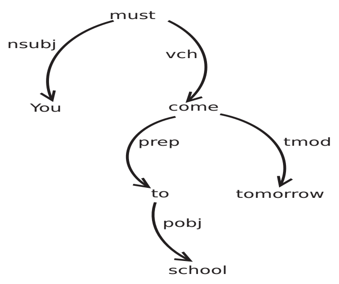

Figure 3: Original Dependency Tree

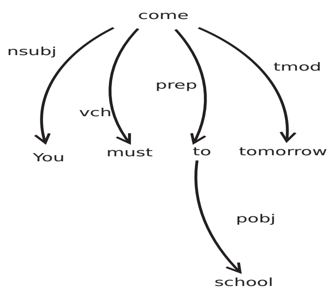

Figure 4: Dependency Tree with *Verb Chain Regularization*

Passive Voice Regularization

Passive Voice Regularization combines active voice and passive voice syntactic structure. For example, even with Verb Chain Regularization "*Michael was defeated by Kobe.*" and "*Kobe defeated Micheal.*" have different syntactic structures. However they have the same meaning. To match the syntactic patterns with passive voice and active voice structure, we introduce Passive Voice Regularization.

Passive Voice Regularization includes two types of changes of the syntactic structure: tagging the ‘real’ subject and the ‘real’ object.

Michael was defeated by Kobe. (3)

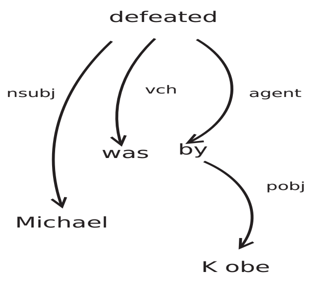

Figure 5: Original Dependency Tree with Verb Chain Regularization

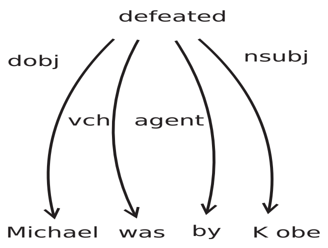

Figure 6: Dependency Tree with *Passive Voice Regularization*

In the sentence above, we regularize the syntactic structure by transforming “defeated-(agent)-by-(pobj)-Kobe” to “defeated-(nsubj)-Kobe”. This transformation tags the “real” subject. On the other hand, the dependency relation “defeated-(nsubj)-Micheal” is changed to “defeated-(dobj)-Micheal”. This tags the “real” object.

Relative Clause Regularization

Unlike other dependency regularizations, Relative Clause Regularization add another dependency relation to the original dependency structure. The new directed graph representing the dependency structure may not be acyclic. Relative Clause Regularization considers two types of relative clauses:

1. Regular Relative Clause

The boy whom I saw yesterday went home . (4)

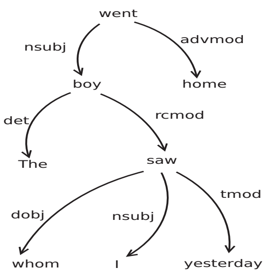

Figure 7: Original Dependency Tree

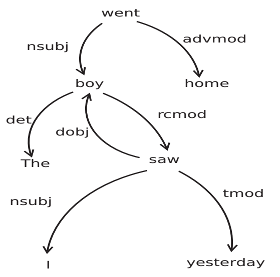

Figure 8: Dependency Tree with Regular Relative Clause Regularization

2. Reduced Relative Clause

I like the boy playing basketball. (5)

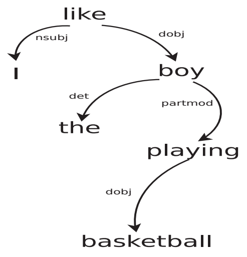

Figure 9: Original Dependency Tree

System Description

Jet, the Java Extraction Toolkit[2], provides a set of NLP components which can be combined to create information extraction systems. AceJet[3] is a sub-system of Jet to extract

[2]http://cs.nyu.edu/grishman/jet/jet.html

[3]http://cs.nyu.edu/grishman/jet/guide/ACEutilities.html

Methods	P	R	F
Sentence-level in (Ji and Grishman 2011)	67.6	53.5	59.7
MaxEnt classifier with local features in (Li, Ji, and Huang 2013)	74.5	59.1	65.9
Joint beam search with local features in (Li, Ji, and Huang 2013)	73.7	59.3	65.7
Joint beam search with local and global features in (Li, Ji, and Huang 2013)	73.7	62.3	67.5
Cross-entity in (Ji and Grishman 2011) †	72.9	64.3	68.3
AceJet baseline system	66.4	69.2	67.7
AceJet with dependency regularization	**68.2**	**69.2**	**68.7**

Table 1: Performance comparison (%) with the state-of-the-art systems. † beyond sentence level.

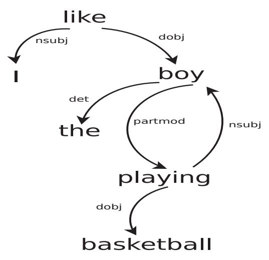

Figure 10: Dependency Tree with Reduced Relative Clause Regularization

the types of information (entities, relations, and events) annotated on the ACE corpora. The AceJet Event Extraction framework is a combination of a pattern-based system and feature-based system.

Training proceeds in three passes over the annotated training corpus. *Pass 1* collects all the event patterns, where a pattern consists of a trigger and a set of arguments along with the syntactic path from the trigger to each argument, and both the dependency path and the linear sequence path (a series of noun chunks and words) are recorded. *Pass 2* records the frequency with which each pattern is associated with an event type – the 'event score'. *Pass 3* treats the event score as a feature, combines it with a small number of other features and trains a maximum entropy model.

At test time, to classify a candidate trigger (any word which has appeared at least once as a trigger in the training corpus) the tagger finds the best match between an event pattern and the input sentence and computes an event score. This score, along with other features, serves as input to the maximum entropy model to make the final ED prediction.

We incorporate the proposed *Dependency Regularization* techniques based on the AceJet baseline system to improve the system performance.

Experiment

In this subsection, we will introduce the evaluation dataset, compare the performance of applying dependency regularization methods with other state-of-the-art systems, and discuss the contributions of these different dependency regularization rules.

Dataset

We used the ACE 2005 corpus as our testbed. For comparison, we used the same test set with 40 newswire articles (672 sentences) as in (Ji and Grishman 2008; Liao and Grishman 2010) for the experiments, and randomly selected 30 other documents (863 sentences) from different genres as the development set. The remaining 529 documents (14,840 sentences) are used for training. We also did experiments using the entire corpus in 10-fold cross-validation.

Following previous work (Ji and Grishman 2008; Liao and Grishman 2010; Ji and Grishman 2011; Li, Ji, and Huang 2013), a trigger candidate is counted as correct if its event subtype and offsets match those of a reference trigger. The ACE 2005 corpus has 33 event subtypes that, along with one class "*None*" for the non-trigger tokens, constitutes a 34-class classification problem. Finally we use Precision (P), Recall (R), and F-measure (F1) to evaluate the overall performance.

Table 1 presents the overall performance of the systems with gold-standard entity mention and type information. We can see that our system with dependency regularizations can improve the performance over our baseline setting, and also advances the current state-of-the-art systems.

Overall Performance

Tables 2 and 3 show the effects of individual regularization rules: Table 2 using the standard 40-document test set, Table 3 using cross-validation. (To apply passive voice regularization, we need to first implement verb chain regularization. Therefore we show the performance with the combination of verb chain and passive voice regularization.) We expect that the cross-validation results are more indicative since they involve a larger test sample. (The absolute scores are lower because the larger test sample involves multiple text genres.) In addition to trigger scores, these tables report accuracy in finding event arguments and associating roles with these arguments.

Table 3 shows that verb chain regularization and passive voice regularization do help improve the performance of event extraction. However, relative clause regularization seems not to improve the system. This difference in effect can be roughly understood in terms of the event identification process described earlier. If a trigger word is associated

Regularization	Trigger	Argument	Role
original	67.75	44.06	39.85
vch	68.51	**44.77**	40.61
vch & pv	68.51	44.65	**40.87**
rc	67.82	43.75	39.66
rc & vch & pv	**68.73**	44.57	40.79

Table 2: Event Extraction performance (%) with different dependency regularizations, where original – original dependency parse output without regularization, *vch* – verb chain regularization, *pv* – passive voice regularization, and *rc* – relative clause regularization.

Regularization	Trigger	Argument	Role
original	54.505	40.913	37.363
vch	54.868	41.461	37.570
vch & pv	**55.116**	**41.647**	**38.186**
rc	54.687	40.866	37.300
rc & vch & pv	55.102	41.439	37.900

Table 3: Event Extraction Performance(%) with 10-fold cross validation, where original – original dependency parse output without regularization, *vch* – verb chain regularization, *pv* – passive voice regularization, and *rc* – relative clause regularization.

in the training corpus with two or more event types, including *None*, it will give rise to multiple event patterns. At test time, to disambiguate a trigger word, we will seek the best match to an event pattern, including a match between the dependency paths. If the training example has a modal or auxiliary while the test sentence has a tensed verb, the paths will be different and no match will be possible. *vch* regularization maps these to a common structure, enabling dependency paths to be aligned and some triggers to be disambiguated. Similarly if the training example is passive and the test example active, the paths cannot be correctly aligned; passive regularization makes a correct alignment possible. In contrast, if the training example involves a full clause and the test example a relative clause, generally one of the arguments (subject or object) can be aligned, which is generally sufficient to disambiguate the trigger. The relative clause regularization may permit an additional argument to be aligned, but this infrequently improves disambiguation.

Table 4 reports on the number of matches between candidate triggers and event patterns, confirming that each of the regularization rules leads to an increase in the number of matches.

Examples of events identified through regularization rules include:

1. With *Verb Chain Regularization*, the sentence "*Taco ball is **appealing**.*" is detected as an APPEAL event, which was ignored in the original framework.
2. With *Passive Voice Regularization*, the sentence "*Thousands of people were **killed** by the army* ." is detected as a DIE event, which was ignored in the original framework.

Upper and Lower Bounds

The AceJet system has a baseline F-score of 67.7% for event detection, which is high and comparable to the current state-of-the-art performance. Based on our investigation, the potential performance of event extraction with dependency regularization ranges from 62.6% to 77.4%, where the upper bound and lower bound correspond to different modifications on top of the original event patterns.

1. **Upper Bound** Since the test data contains many triggers that never appear in the training data, such as the ATTACK event trigger *intifada*, some events cannot be detected solely using the training data. Hence, there is an upper bound for the performance of event detection. We extracted event patterns from the test data with the triggers that appear in the training data. Then we added these patterns to the pattern set extracted from the training data. With the expanded patterns, the F-score of event detection reaches 77.4%, which is considered as the upper bound of the performance with dependency regularization.
2. **Lower Bound** The trigger word itself contains the most indicative information about the event, and is often sufficient to trigger an event. In other words, dependency regularization would not help in this case. For example, "*has died*" is a high-frequency pattern in ACE data, but the verb "*die*" itself is enough to detect the DIE event. Even without dependency regularization, the word "die" may still be identified as a DIE event trigger, because many DIE events are triggered by the verb "*die*" in the training data. Other syntactic information would not help much either here, compared to the dominant contribution of "*die*" word itself. We removed all the roles from the patterns and only kept the triggers, which resulted in an F-score of 62.6%. We consider it as the lower bound of the event detection performance with dependency regularization.

Related Work

Although there have been quite a few distinct designs for event extraction systems, most are loosely based on using patterns to detect instances of events, where the patterns consist of a predicate, *event trigger*, and constraints on its local

Regularization	Matched Patterns	Increase	Decrease
original	6274	–	–
vch	6815	143	5 (compared to orig)
vch & pv	6862	18	4 (compared to vch)
rc	6367	45	1 (compared to orig)

Table 4: Matched Event Patterns of the candidates *pv* – passive voice regularization, and *rc* – relative clause regularization. *Increase* – the number of candidates matching more patterns, *Decrease* – the number of candidates matching less patterns

syntactic context. The constraints may involve specific lexical items or semantic classes. Some recent studies use high-level information to aid local event extraction systems. For example, (Finkel, Grenager, and Manning 2005), (Maslennikov and seng Chua 2007), (Ji and Grishman 2008) and (Patwardhan and Riloff 2007) tried to use discourse, document, or cross-document information to improve information extraction. Other research extends these approaches by introducing cross-event information to enhance the performance of multi-event-type extraction systems. (Liao and Grishman 2010) use information about other types of events to make predictions or resolve ambiguities regarding a given event. (Li, Ji, and Huang 2013) implements a joint model via structured prediction with cross-event features.

Event extraction systems have used patterns and features based on a range of linguistic representations. For example, (Miwa et al. 2014) used both a deep analysis and a dependency parse. The original NYU system for the 2005 ACE evaluation (Grishman, Westbrook, and Meyers 2005) incorporated GLARF, a representation which captured both notions of transparency and verb-nominalization correspondences.[4] However, assessment of the impact of individual regularizations has been limited; this prompted the investigation reported here.

Dependency Regularization has been utilized before to improve the performance of detecting event triggers with specific types. (Cao, Li, and Grishman 2015) used 3 types of dependency regularizations: Verb Chain Regularization, Transparent Regularization, and Nominalization Regularization. (Cao et al. 2015) used active learning to fill gaps in the ACE event training data, and (Li et al. 2015) improved the event detection performance by exploiting the semantic knowledge encoded in Abstract Meaning Representation.

Conclusion

In this paper we have proposed several *Dependency Regularization* steps to improve the performance of the *Event Extraction* framework, including *Passive Voice Regularization* and *Relative Clause Regularization*. The experimental results have demonstrated the effectiveness of these techniques, which has helped our pattern-based trigger detection system achieve 68.7% F-measure (with 1.1% absolute improvement over the baseline). In addition, this dependency regularization technique can also help enhance the performance of our event argument detection and role labeling by around 0.7% and 1.0% absolute improvement.

[4]The official evaluations were made with a complex *value* metric and so are hard to compare with more recent results.

References

Cao, K.; Li, X.; Fan, M.; and Grishman, R. 2015. Improving event detection with active learning. In *Proceedings of RANLP*.

Cao, K.; Li, X.; and Grishman, R. 2015. Improving event detection with dependency regularization. In *Proceedings of RANLP*.

Finkel, J. R.; Grenager, T.; and Manning, C. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Grishman, R.; Westbrook, D.; and Meyers, A. 2005. NYU's English ACE 2005 system description. In *Proceedings of the ACE 2005 Evaluation Workshop*.

Ji, H., and Grishman, R. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL*.

Ji, H., and Grishman, R. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL*.

Li, X.; Nguyen, H. T.; Cao, K.; and Grishman, R. 2015. Improving event detection with abstract meaning representation. In *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS 2015)*.

Li, Q.; Ji, H.; and Huang, L. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*.

Liao, S., and Grishman, R. 2010. Using document level cross-event inference to improve event extraction. In *Proceedings of ACL*.

Maslennikov, M., and seng Chua, T. 2007. A multi-resolution framework for information extraction from free text. In *Proceedings of ACL*.

Miwa, M.; Thompson, P.; Korkontzelos, I.; and Ananiadou, S. 2014. Comparable study of event extraction in newswire and biomedical domains. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*.

Patwardhan, S., and Riloff, E. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of EMNLP*.

Tratz, S., and Hovy, E. 2011. Fast, effective, non-projective, semantically-enriched parser. In *Proceedings of EMNLP*.