Improving Event Detection with Dependency Regularization

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

Abstract

Event Detection (ED) is an Information Extraction task which involves identifying instances of specified types of events in text. Most recent research on Event Detection 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 ED performance. In this paper, we demonstrate the effectiveness of *Dependency Regularization* techniques to generalize the patterns extracted from the training data to boost ED performance. The experimental results on the ACE 2005 corpus show that our pattern-based system with the expanded patterns can achieve 70.49% (with 2.57% absolute improvement) F-measure over the baseline, which advances the state-of-the-art for such systems.

1 Introduction

Event Detection (ED) involves identifying instances of specified types of events 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 and classifying them into specific types. For instance, according to the ACE 2005 annotation guidelines[1], in the sentence "*She was **killed** in an automobile accident yesterday*", an event detection system should be able to recognize the word "*killed*" as a trigger for the event DIE. 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. ED is a crucial component in the overall Event Extraction task, which also requires event argument identification and argument role labeling.

Most recent research on Automatic Content Extraction (ACE) Event Detection task relies on pattern-based or feature-based approaches to building classifiers for event trigger 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 ED 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 70.49% (with 2.57% absolute improvement) F-measure over the baseline, which is an advance over the state-of-the-art systems.

The paper is organized as follows: In Section 2, we will describe the role of dependency analysis in event detection and how dependency regularization methods can improve ED performance. We will describe our ED systems including the baseline and enhanced system utilizing dependency regularization in Section 3, and present experi-

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

Proceedings of Recent Advances in Natural Language Processing, pages 78–83,
Hissar, Bulgaria, Sep 7–9 2015.

mental results in Section 4. We will discuss related work in Section 5, and Section 6 will conclude this work and list our research directions.

2 Dependency Regularization

The ACE 2005 Event Guidelines specify a set of 33 types of events; 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 and 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.

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 adopt a dual approach using a set of *dependency regularization* rules: in some cases we will transform the syntactic structure of the input to reduce variation; in other cases we will expand the patterns to handle a wider variety of input.

We describe here three of the regularization rules we use:

1. verb chain regularization
2. transparent word regularization
3. nominalization regularization

2.1 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)

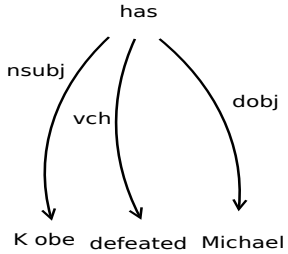

Figure 1: Original Dependency Tree

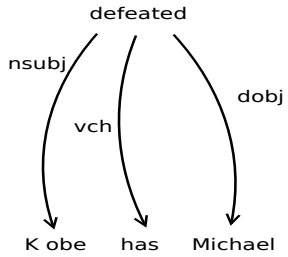

Figure 2: Dependency Tree with *Verb Chain Regularization*

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:

You must come to school tomorrow . (2)

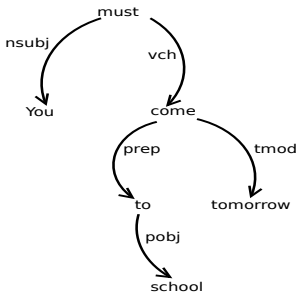

Figure 3: Original Dependency Tree

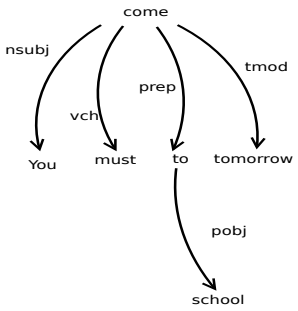

Figure 4: Dependency Tree with *Verb Chain Regularization*

2.2 Transparent Word Regularization

Some words, such as those expressing quantities, are semantically 'transparent': they take on the semantic type of their object. For purposes of determining event types, we want to 'look through' such words in the dependency parse. We do so by restructuring the tree. This is one of the most useful dependency regularization rules, since the dependency path is shortened and the head should reach the "real" dependent directly.

The army killed thousands of people . (3)

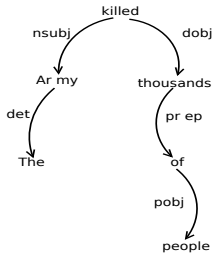

Figure 5: Original Dependency Tree

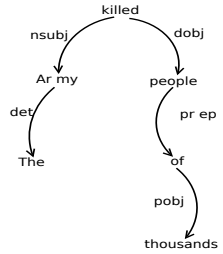

Figure 6: Dependency Tree with *Transparent Regularization*

In this case the semantic type of the object of the verb "*kill*" is determined by the word "*people*" instead of the word "*thousands*". Especially in the pattern-based framework, this kind of improvement helps substantially in finding the roles of the events.

2.3 Nominalization Regularization

Most types of events can be expressed by verbal or nominal constructions. However, in a number of cases the ACE training corpus includes the verbal construction but not the corresponding nominal one. We addressed this problem by automatically generating the nominal pattern from the verbal one. (The reverse case, with only a nominal pattern, was less frequent.)

Nomlex (NOMinalization LEXicon) is a dictionary of English nominalizations developed at New York University under the direction of Catherine Macleod. NOMLEX seeks not only to describe the allowed complements for a nominalization, but also to relate the nominal complements to the arguments of the corresponding verb. Therefore with Nomlex we can expand the patterns evoked

Methods	P	R	F
Sentence-level in (Ji and Grishman, 2011)	67.6	53.5	59.7
MaxEnt classifier with local features in (Li et al., 2013)	74.5	59.1	65.9
Joint beam search with local features in (Li et al., 2013)	73.7	59.3	65.7
Joint beam search with local and global features in (Li et al., 2013)	73.7	62.3	67.5
Cross-entity in (Ji and Grishman, 2011) †	72.9	64.3	68.3
AceJet baseline system	65.4	70.6	67.9
AceJet with dependency regularization	**68.2**	**72.8**	**70.4**

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

by verb triggers to patterns evoked by noun triggers. This translation is based on the correspondence between a verb with its arguments and a nominalization with its arguments.

For example, the sentence "*Microsoft acquired Nokia yesterday*" is an instance of the *Transfer-Ownership* event. "*The acquisition of Nokia from Microsoft was successful yesterday*" is also an event instance of the same type. However, they do not share the same event pattern. Our heuristic methods of dependency regularization transform one pattern into the other.

There are three types of pattern transformations, assigning different roles to the object of the verb. Let us suppose the original sentence is:

IBM appointed Alice Smith as vice president . (4)

Then we would automatically generate additional patterns for:

1. DET-POSS: a possessive determiner.

   Alice Smith's appointment as vice president (5)

2. N-N-MOD: a nominal modifier

   the Alice Smith appointment as vice president (6)

3. PP-OF: object of the preposition

   the appointment of Alice Smith as vice president (7)

In the sentences above , "*Alice Smith*" is the person who gets the job, and the phrase "*vice president*" is Alice's position. Thus the sentences share the same arguments, although the syntactic patterns are different.

3 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 subsystem of Jet to extract 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 path from the trigger to each argument; 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.

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

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

4 Experiment

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

4.1 Data set

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.

Regarding the correctness criteria: following previous work (Ji and Grishman, 2008; Liao and Grishman, 2010; Ji and Grishman, 2011; Li et al., 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 in this work. 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.

4.2 Contributions of different dependency regularizations

Table 2 lists the system performance applying the different dependency regularization rules. The last line shows the performance with the combination of three types of Nomlex pattern expansion.

Dependency Regularization could help match patterns that failed in the original framework. For example,

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 *Transparent Regularization*, the sentence "*The army **killed** thousands of people.*" is detected as a DIE event, which was ignored in the original framework.

3. With *Nomlex Regularization*, the sentence "*The **acquisition** of Banco Zaragozano...*" is detected as a TRANSFER-OWNERSHIP event, which was ignored in the original framework. This is because all the relevant sentences in the training data use the same trigger "*acquire*".

5 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 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 et al. (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 et al. (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 et al., 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.

6 Conclusion and Future Work

In this paper we have proposed several *Dependency Regularization* steps to improve the perfor-

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

Regularization	Recall	Precision	F-measure
original	65.45	70.59	67.92
vch	66.82	70.84	68.77
transp	65.68	71.18	68.32
vch & transp	67.27	71.50	69.32
vch & transp & Nomlex	**68.18**	**72.82**	**70.42**

Table 2: Trigger identification performance (%) with different dependency regularizations, where original – original dependency parse output without regularization, *vch* – verb chain regularization, *transp* – transparent regularization, and *Nomlex* – Nomlex regularization.

mance of the *Event Detection* framework, including *Verb Chain Regularization*, *Transparent Regularization*, and *Nomlex Regularization*. The experimental results have demonstrated the effectiveness of these techniques, which has helped our pattern-based system achieve 70.49% (with 2.57% absolute improvement) F-measure over the baseline, which significantly advances the state-of-the-art systems.

Dependency regularization is only one of the measures we can take to improve performance. The training corpus cannot include all possible trigger words or all the senses of the triggers it does include. Simply enlarging the training corpus by sequential annotation would yield small gain at a large cost. In parallel work we have shown that carefully targeted active learning of new triggers and senses can produce significant improvement in event detection at modest cost.

References

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of ACL*.

Ralph Grishman, David Westbrook, and Adam Meyers. 2005. NYU's English ACE 2005 system description. In *Proceedings of the ACE 2005 Evaluation Workshop*.

Heng Ji and Ralph Grishman. 2008. Refining event extraction through cross-document inference. In *Proceedings of ACL*.

Heng Ji and Ralph Grishman. 2011. Using cross-entity inference to improve event extraction. In *Proceedings of ACL*.

Qi Li, Heng Ji, and Liang Huang. 2013. Joint event extraction via structured prediction with global features. In *Proceedings of ACL*.

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

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

Makoto Miwa, Paul Thompson, Ioannis Korkontzelos, and Sophia Ananiadou. 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*.

Siddharth Patwardhan and Ellen Riloff. 2007. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of EMNLP*.

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