IMPROVING EVENT EXTRACTION : CASTING A WIDER NET

by

Kai Cao

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January, 2017

Professor Ralph Grishman

Dedication

To my parents, grandparents and all the family members.

Acknowledgments

Studying for a PhD and living in a foreign country has been a valuable and exciting experience for me. I could not have successfully graduate without the advice, support and influence of many colleagues, friends and family.

Firstly and most importantly, I am eternally grateful to my advisor, Prof. Ralph Grishman, for his generous help and continuous guidance throughout my graduate study. I have always tried to learn from Ralph's enthusiasm in pursuing all kinds of new research problems. His professional expertise with a sense of humor and positive attitude inspired me a lot. I will be forever grateful for his help and proud to be his student.

I am very honored to have the great professors in my doctoral committee: Prof. Ralph Grishman, Prof. Heng Ji, Prof. Ernest Davis, Prof. Adam Meyers and Prof. Wei Xu, who took the most valuable time in reading my thesis and providing me with extensive comments. I would like to thank all of them for their insightful suggestions for my thesis.

I also feel very lucky to be a member of Proteus group in NYU during my graduate NLP research. I am greatly honored to work with such wonderful team members. I would like to say many thanks to Prof. Adam Meyers for his generous help and guidance. It is also a great honor for me to work with Dr. Bonan Min, Dr. Yifan He, Dr. Wei Xu, Dr. Maria Pershina, Thien Huu Nguyen, Lisheng Fu, Miao Fan, Jiawei Wu and Angus Grieve-Smith.

Last, but not least, I would like to thank my parents and all the family members for their selfless support of my PhD study and unfailing faith in me.

Abstract

Information extraction is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. One facet of information extraction is event extraction (EE): identifying instances of selected types of events appearing in natural language text. For each instance, EE should identify the type of the event, the event trigger (the word or phrase which evokes the event), the participants in the event, and (where possible) the time and place of the event.

One EE task was defined and intensively studied as part of the ACE (Automatic Content Extraction) research program. The 2005 ACE EE task involved 8 types and 33 subtypes of events. For instance, given the sentence “She was killed by an automobile yesterday.”, an EE system should be able to recognize the word “killed” as a trigger for an event of subtype 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 types of events in different contexts.

To support the development and evaluation of ACE EE systems, the Linguistic Data Consortium annotated a text corpus (consisting primarily of news articles) with information on the events mentioned. This corpus was widely used to train ACE EE systems. However, the event instances in the ACE corpus are not evenly distributed, and so some frequent expressions involving ACE events do not appear in the training data, adversely affecting performance.

This thesis presents several strategies for improving the performance of EE. We first demonstrate the effectiveness of two types of linguistic analysis – dependency regularization and Abstract Meaning Representation – in boosting EE performance. Next we show the benefit of an active learning strategy in which a person is asked to judge a limited number of phrases which may be event triggers. Finally we report the impact of combining our baseline system with event patterns from a system developed for a different EE task (the TABARI program). This step contains expert-level patterns generated by other research groups. Because the information received is complicated and quite different from the original corpus (ACE), the integration of this information requires more complex processing.

Table of contents

List of Figures

List of Tables

Chapter 1

Introduction

Information is plentiful and readily available, from the Internet, news services, media, etc.. Extracting the critical nuggets that matter to business or to national security is a cognitively demanding and time consuming task. Intelligence and business analysts spend many hours poring over endless streams of text documents pulling out references to entities of interest (people, locations, organizations) as well as their relationships as reported in text (Liu, 2009). However, if we can automatically identify such information, we can eliminate or at least reduce the human labor and speed up the process. Information Extraction (IE) automatically extracts structured data from text documents. Normally information extraction contains three levels of extraction tasks: entity extraction identifies all the useful snippets in text, such as people, locations and organizations; relation extraction identifies all the binary relations between entities, and event extraction identifies multi-way relations among entities.

1.1 Named Entity Recognition

Named Entity Recognition (NER) is the core annotation task of ACE, providing the foundation for all remaining tasks. This ACE task identifies seven types of entities: Person, Organization, Location, Facility, Weapon, Vehicle and Geo-Political Entity (GPE). Each type is further divided into subtypes. Annotators tag all mentions of each entity within a document, whether named, nominal (common noun) or pronominal. For every mention, the annotator identifies the maximal extent of the string that represents the entity, and labels the head of each mention. For example, in processing the sentence "M. Smith likes fishing", named entity detection would denote detecting that the phrase "M. Smith" does refer to a person, but without necessarily having (or using) any knowledge about a certain M. Smith who is (or, "might be") the specific person whom that sentence is talking about.

1.2 Relation Extraction

Relation Detection and Recognition (RDR) involves the identification of relations between entities. As RDR is considered separate from the event extraction task, we do not provide more details in this thesis.

1.3 Event Extraction

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, in the sentence "*[**She**] was <u>**killed**</u> 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.

Event Extraction 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. Besides, as the event instances in the ACE corpus are not evenly distributed, some frequent expressions involving ACE events do not appear in the training data, adversely affecting performance. Moreover, some event triggers in the test data never appear in the training data. In our research, we studied the following four approaches for improving event extraction:

1. Dependency Regularization
2. Abstract Meaning Representation
3. Active Learning
4. Expert-level Patterns

1.4 Structure of the thesis

This thesis is organized as follows: Chapter 2 introduces the ACE corpus and ACE Events. Chapter 3 summarizes the related work on event extraction. Chapter 4 introduces NYU's event extraction baseline system, which is based on the combination of pattern-based and feature-based models.

Chapter 5 proposes several *Dependency Regularization* steps to improve the performance of the *Event Extraction* framework, including *Verb Chain Regularization*, *Passive Voice Regularization*, *Relative Clause Regularization*, *Transparent Regularization*, *Nominalization Regularization*, *Subject Control Regularization* and *Subject Raising Regularization*. The experimental results have demonstrated the effectiveness of these techniques, which has helped our pattern-based trigger detection system achieve 70.4% F-measure (with 2.5% absolute improvement over the baseline).

Chapter 6 demonstrates that Abstract Meaning Representation can capture deeper contexts of trigger words in this task, and the experimental results show that adding AMR features on top of the traditional features can achieve 67.8% in F-measure with 2.1% [1]absolute improvement over the baseline features. We show that AMR enables event extraction performance to become comparable to the state-of-the-art approaches.

Chapter 7 demonstrates the effectiveness of active learning to import more patterns extracted from external corpora to boost Event Detection performance. Since these newly added patterns may never appear in the training data, they

1. The baseline of feature-based systems is 65.7%, while the baseline of pattern-based systems is 67.9%

can complement the patterns generated from the original training data to enhance event extraction performance. The experimental results show that our pattern-based system with the expanded patterns can achieve 70.4% (with 2.5% absolute improvement) F-measure over the baseline, an advance over current state-of-the-art systems.

Chapter 8 demonstrates the effectiveness of systematically importing expert-level patterns from TABARI to boost EE performance. The experimental results demonstrate that our pattern-based system with the expanded patterns can achieve 69.8% (with 1.9% absolute improvement) F-measure over the baseline, an advance over current state-of-the-art systems.

Chapter 9 evaluates the combination of different approaches to improve the performance of *Event Extraction* systems, including dependency regularization, active learning and expert-level patterns.

Chapter 10 concludes the thesis as a whole, pointing out some possible directions for future work.

Chapter 2

ACE & ACE Events

Event extraction (also referred to as “scenario template” extraction) involves the identification in free text of instances of a particular type of event, and the identification of the arguments of each such event.

There are two event extraction tasks that are widely investigated: one is the MUC event extraction tasks, including MUC-3/4 on Latin American terrorist incidents (MUC 1991; MUC 1992), and MUC-6 on executive succession (MUC 1995); the other is the ACE 2005 (33 event types covering the most common events of national and international news) (ACE 2005). In this thesis, we focus on the studies on ACE event types, and all of the experiments are reported on the ACE 2005 evaluation.

2.1 ACE Corpus

ACE (Automatic Content Extraction) began in 2000 after MUC. "The objective of the ACE program is to develop automatic content extraction technology to support automatic processing of human language in text form from a variety of sources (such as newswire, broadcast conversation, and weblogs). ACE technology R&D is aimed at supporting various classification, filtering, and selection applications by extracting and representing language content (i.e., the meaning conveyed by the data). Thus the ACE program requires the development of technologies that automatically detect and characterize this meaning." [1] Unlike MUC data, which was primarily extracted from newswire, ACE also includes data from manually and automatically transcribed broadcast news, Internet blogs, etc., thus, the text is often of poor quality when compared to MUC data. The ACE research objectives are viewed as the detection and characterization of Entities, Relations and Events.

The 2005 ACE corpus contains 599 files; in most studies of event extraction, 529 files (14,840 sentences) are taken as training data, while 30 files (863 sentences) are used as development data and the remaining 40 files (672 sentences) are the test data. There are 440 events in the test data while there are more than 5000 events in the training data. Before understanding ACE event extraction, we start with an introduction to ACE events.

1. `http://www.nist.gov/speech/tests/ace/`

2.2 ACE Events

ACE event contains 8 types and 33 subtypes, which are shown as below:

1. Life: Life events are mainly about the milestones of a person's life.

    a) Be-Born: A Be-Born Event occurs whenever a person is given birth to.

    b) Marry: Marry Events are official Events, where two people are married under the legal definition.

    c) Divorce: A Divorce Event occurs whenever two people are officially divorced under the legal definition of divorce.

    d) Injure: An Injure Event occurs whenever a Person Entity experiences physical harm.

    e) Die: A Die Event occurs whenever the life of a Person Entity ends. DIE Events can be accidental, intentional or self-inflicted.

2. Movement: There is only one subtype of Movement Events:

    a) Transport: A Transport Event occurs when a weapon, vehicle or a person is moved from one place to another.

3. Transaction:

    a) Transfer-Ownership: Transfer-Ownership Events refer to the buying, selling, loaning, borrowing, giving, or receiving of artifacts or organizations.

   b) Transfer-Money: Transfer-Money Events refer to the giving, receiving, borrowing, or lending money when it is not in the context of purchasing something.

4. Business:

   a) Start-Org: A Start-Org Event occurs whenever a new organization is created.

   b) Merge-Org: Merge-Org means two organizations come together to form a new one.

   c) Declare-Bankruptcy: A Declare-Bankruptcy Event will occur whenever an Entity officially requests legal protection from debt collection due to an extremely negative balance sheet.

   d) End-Org: An End-Org Event occurs whenever an Organization ceases to exist.

5. Conflict

   a) Attack: An Attack Event is defined as a violent physical act causing harm or damage.

   b) Demonstrate: A Demonstrate Event occurs whenever a large number of people come together in a public area to protest or demand some sort of official action.

6. Contact:

   a) Meet: A Meet Event occurs whenever two or more Entities come together at a single location and interact with one another face-to-face.
   b) Phone-Write: A Phone-Write Event occurs when two or more people directly engage in discussion which does not take place face-to-face.

7. Personnel:
   a) Start-Position: A Start-Position Event occurs whenever a person begins working for (or changes offices within) an organization.
   b) End-Position: A Start-Position Event occurs whenever a person stops working for (or changes offices within) an organization.
   c) Nominate: A nominate Event occurs whenever a person is proposed for a Start-Position Event by the appropriate person, through official channels.
   d) Elect: An Elect Event occurs whenever a candidate wins an election designed to determine the person argument of a Start-Position Event.

8. Justice: Justice Events are classified as the 13 types below. Each event can be clearly explained by its own name of the event subtype.
   a) Arrest-Jail
   b) Release-Parole
   c) Trial-Hearing
   d) Charge-Indict
   e) Sue

f) Convict

g) Sentence

h) Fine

i) Execute

j) Extradite

k) Acquit

l) Appeal

m) Pardon

Each event takes one or more arguments. Most arguments are entities, which are references in the text to people, organizations, locations, facilities, weapons, or vehicles.

2.3 Evaluation

The evaluation is based on event mentions. An *Event Mention* is a phrase or sentence within which an event is described, including trigger and arguments. An event mention must have one and only one trigger, and can have an arbitrary number of arguments. We use the precision, recall, and F-measure standard metrics to evaluate the system performance, which are defined as follows:

$$Precision = \frac{\mid System\ Samples\ \cap\ Key\ Samples \mid}{\mid System\ Samples \mid} \tag{2.1}$$

$$Recall = \frac{\mid System\ Samples\ \cap\ Key\ Samples \mid}{\mid Key\ Samples \mid} \tag{2.2}$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.3}$$

Since an ACE event contains one trigger and an arbitrary number of arguments, its structure is somewhat complicated and it is hard to evaluate it as a whole. As a result, we prefer to examine the system performance at three levels, the trigger classification, the argument identification and the argument classification. The trigger classification assesses how well the system can detect events and their types; argument identification assesses how well the system finds arguments of the events; argument classification assesses how well the system assign roles for the arguments. The three metrics define a correct instance as one matching the key with respect to the following elements:

Evaluation Metric	Matched Elements
Trigger Labeling	Event type and subtype Trigger start offset Trigger end offset
Argument Identification	Event type and subtype Argument head start offset Argument head end offset
Argument Classification	Event type and subtype Argument head start offset Argument head end offset Argument role

Table 2.1: The elements that need to be matched for each evaluation metric

Chapter 3

Related Work

Prior work in event extraction is mainly of two types: a pattern-based framework or feature-based system. The main work to improve the performance has focused largely on either improving the pattern-matching kernel or adding new reasonable features.

3.1 Feature-based Systems

Most event extraction frameworks are feature-based systems. Some of the feature-based systems are based on phrase or sentence level extraction. Several recent studies use high-level information to aid local event extraction systems. For example, Finkel et al. (2005), Maslennikov and Chua (2007), Ji and Grishman (2008), Patwardhan and Riloff (2007) and Hong et al. (2011) tried to use discourse, document, cross-document or cross-entity 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. Cross-event information plays a significant role because:

1. Some events co-occur frequently, while other events do not: For example, Attack, Die,and Injure events occur together very frequently, while Attack and Marry are less likely to co-occur.

2. Typical relations among the arguments of different types of events would be useful in predicting information to be extracted. For example, the Victim of a Die event is probably the Target of the Attack event.

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. Concretely, they use structured perceptron with inexact search to jointly extract triggers and arguments that co-occur in the same sentence.

Neural Networks have been applied to improve event extraction systems. Nguyen and Grishman (2015) studies the event detection problem using convolutional neural networks (CNNs) that overcome the two fundamental limitations of the traditional feature-based approaches to this task: complicated feature engineering for rich feature sets and error propagation from the preceding stages which generate these features. Nguyen et al. (2016) proposes to do event extraction in a joint framework with bidirectional recurrent neural networks, thereby benefiting from the advantages of the two models as well as addressing issues inherent in the existing approaches. They systematically investigate different memory features for the joint model. Nguyen and Grishman (2016) proposes to improve the previous CNN models (Nguyen and Grishman, 2015) for event detection by operating the convolution on all possible non-consecutive k-grams in the sentences. They aggregate the resulting convolution scores via the max-pooling function to unveil the most important non-consecutive k-grams for event detection. Feng et al. (2016) develops a hybrid neural network to capture both sequence and chunk information from specific contexts, and use them to train an event detector for multiple languages without any manually encoded features.

3.2 Pattern-based Systems

Although there have been quite a few distinct designs for event extraction systems, most pattern-based systems 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.

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.[1] However, assessment of the impact of individual regularizations has been limited; this prompted the investigation reported here.

There have been several efforts over the past decade to develop semi-supervised

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

methods for learning pattern sets. One thread began with Riloff's observation that patterns occurring with substantially higher frequency in relevant documents than in irrelevant documents are likely to be good extraction patterns (Riloff, 1996). Sudo et al. (2003) sorted relevant from irrelevant documents using a topic description and information retrieval engine. Yangarber (2003) and Yangarber et al. (2000) developed a bootstrapping approach, starting with some seed patterns, using these patterns to identify some relevant documents, using these documents to identify additional patterns, etc. This approach was further refined in Surdeanu et al. (2006), which explored alternative pattern ranking strategies. An alternative approach was adopted in Stevenson and Greenwood (2005), which used Wordnet-based similarity to expand an initial set of event patterns. Huang and Riloff (2012) developed a bootstrapping system to discover new triggers with selected roles. For example, the word "*sniper*" is very likely to be the *agent* of a *Die* event. Bronstein et al. (2015) takes the example trigger terms mentioned in the guidelines as seeds, and then applies an event-independent similarity-based classifier for trigger labeling.

3.3 Performance

Methods	P	R	F_1
Sentence-level in Hong et al. (2011)	67.6	53.5	59.7
MaxEnt classifier with local features in Li et al. (2013)	74.5	59.1	65.9
JBS with local features in Li et al. (2013)	73.7	59.3	65.7
JBS with local and global features in Li et al. (2013)	73.7	62.3	67.5
Cross-event in Liao and Grishman (2010) †	68.7	68.9	68.8
Cross-entity in Hong et al. (2011) †	72.9	64.3	68.3
CNNs in Nguyen and Grishman (2015)	71.8	66.4	69.0
NC-CNNs in Nguyen and Grishman (2016)			71.3
Hybrid Neural Networks in Feng et al. (2016)	84.6	64.9	73.4
Seed-based in Bronstein et al. (2015)	80.6	67.1	73.2

Table 3.1: Performance (%) comparison with the state-of-the-art systems, where JBS – Joint Beam Search. † beyond sentence level.

Chapter 4

The Baseline System

Chapters 5, 7 and 8 try to improve the performance of the same EE system – AceJet. Therefore, before talking about the new approaches used to improve event extraction, we need to start with the introduction of Jet and AceJet.

Jet, the Java Extraction Toolkit[1], provides a set of NLP components which can be combined to create information extraction systems. AceJet[2] is a sub-system 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 syntactic path from the trigger to each argument, and both the dependency path and the linear sequence path (a series of noun chunks

1. `http://cs.nyu.edu/grishman/jet/jet.html`
2. `http://cs.nyu.edu/grishman/jet/guide/ACEutilities.html`

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 event extraction prediction.

The following three maxent models are used as the feature-based part of the system:

1. **Reportable-Event Classifier (Trigger Classifier)**

   Given a potential trigger, an event type, and a set of arguments, it is a binary classifier determining whether there is a reportable event mention.

2. **Argument Classifier**

   The argument classifier is a binary classifier that distinguishes arguments of a potential trigger from non-arguments.

3. **Role Classifier**

   The role classifier is a multi-class classifier to assign each argument with its proper role.

Chapter 5

Improving Event Extraction with Dependency Regularization [1]

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.

1. This chapter is mainly adapted from the published papers (Cao et al., 2015b, 2016)

- *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 employ a set of *dependency regularization* rules which transform the syntactic structure of the input to reduce variation.

5.1 Dependency Regularization

Seven dependency regularization rules are used in the experiment:

1. Verb Chain Regularization

2. Passive Voice Regularization

3. Relative Clause Regularization

4. Transparent Regularization

5. Nominalization Regularization

6. Subject Control Regularization

7. Subject Raising Regularization

5.1.1 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. (5.1)

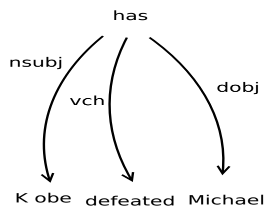


Figure 5.1: Original Dependency Tree without *Verb Chain Regularization* (a)

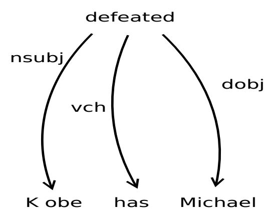


Figure 5.2: Dependency Tree with *Verb Chain Regularization*(a)

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:

$$\textit{You must come to school tomorrow.} \tag{5.2}$$

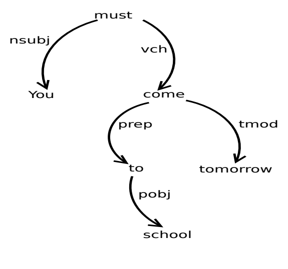


Figure 5.3: Original Dependency Tree without *Verb Chain Regularization*(b)

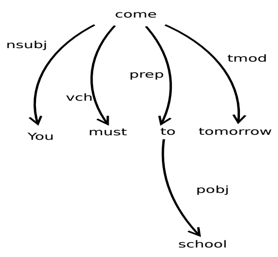


Figure 5.4: Dependency Tree with *Verb Chain Regularization*(b)

5.1.2 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 Michael.*" 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. (5.3)

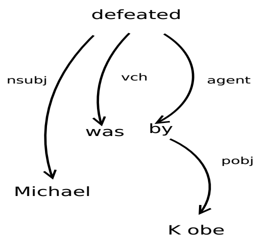


Figure 5.5: Original Dependency Tree with *Verb Chain Regularization*

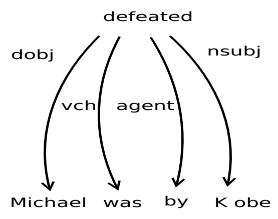


Figure 5.6: Dependency Tree with *Passive Voice Regularization*

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)-Michael" is changed to "defeated-(dobj)-Michael". This tags the "real" object.

5.1.3 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. (5.4)

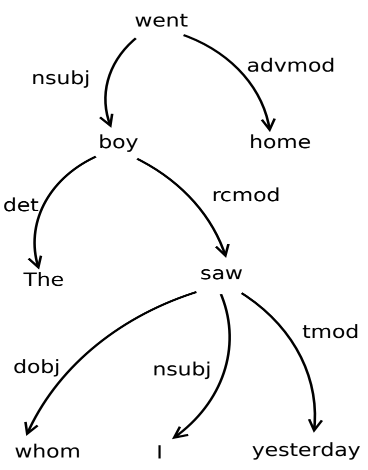


Figure 5.7: Original Dependency Tree without *Regular Relative Clause Regularization*

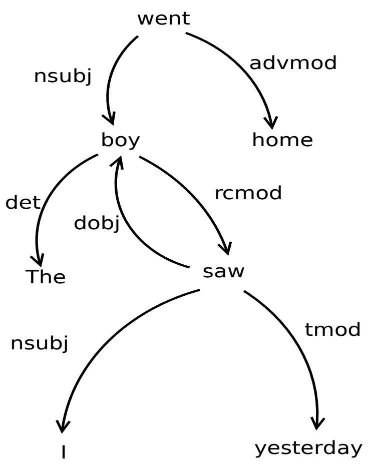


Figure 5.8: Dependency Tree without *Regular Relative Clause Regularization*

2. Reduced Relative Clause

$$\textit{I like the boy playing basketball.} \tag{5.5}$$

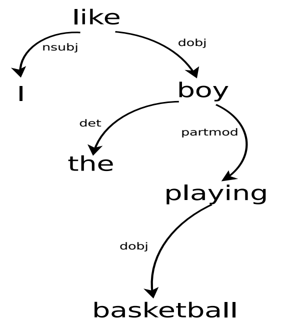


Figure 5.9: Original Dependency Tree without Reduced Relative Clause Regularization

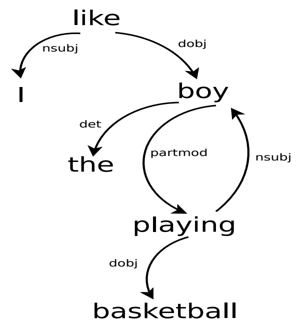


Figure 5.10: Dependency Tree with Reduced Relative Clause Regularization

5.1.4 Transparent 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. (5.6)

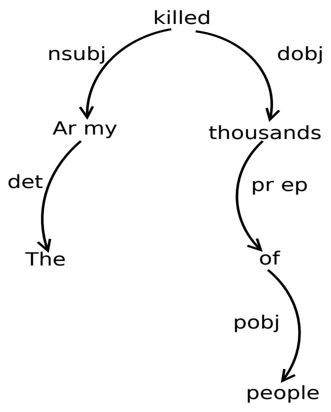


Figure 5.11: Original Dependency Tree without *Transparent Regularization*

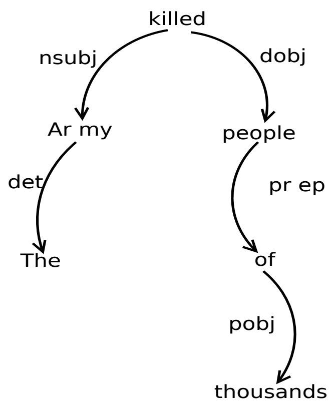


Figure 5.12: Dependency Tree with *Transparent Regularization*

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.

5.1.5 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 [2] 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 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. (5.7)

Then we would automatically generate additional patterns for:

1. DET-POSS: a possessive determiner.

   Alice Smith's appointment as vice president (5.8)

2. N-N-MOD: a nominal modifier

   the Alice Smith appointment as vice president (5.9)

2. `http://nlp.cs.nyu.edu/nomlex/`

3. PP-OF: object of the preposition

$$\textit{the appointment of Alice Smith as vice president} \tag{5.10}$$

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.

5.1.6 Subject Control

Both "Subject Control" and "Subject Raising" are not general dependency regularizations. They are limited to specific lexical items. The classification scheme of words in COMLEX [3] syntax is used to make these determinations. COMLEX Syntax is a monolingual English Dictionary consisting of 38,000 head words intended for use in natural language processing. The dictionary includes entries for approximately 21,000 nouns, 8,000 adjectives and 6,000 verbs, all of which are marked with a rich set of syntactic features and complements. The "Subject Control" list consists of the verbs marked "to-inf-sc" in COMLEX. The "Subject Raising" list consists of the verbs marked "to-inf-rs" in COMLEX.

In linguistics, control is a construction where the understood subject of a given predicate is determined by some expression in context. Control predicates have semantic content; they semantically select their arguments, that is, their appearance strongly influences the nature of the arguments they take. Subject Control is the situation where the verbs/predicates select their subjects. For example, the word "try" restricts the subject:

3. `http://nlp.cs.nyu.edu/comlex/`

1. John tried to fall
2. John tried to make sense
3. *The idea tried to fall
4. *The rock tried to make sense

Obviously sentences 3 and 4 are meaningless because the “idea” and “rock” cannot “try”. These restrictions are part of the evidence that there are semantic relations (agent, etc.) between the subject of some verbs (“try”) and the lower predicates (“fall”). Since both the predicate(“try”) and the lower predicate (“fall”) restrict the subject, we add a new syntactic relation between the lower predicate and the subject. This heuristic comes from the same intuition as transparent regularization: Link the “real” syntactic relations. Subject control would link the roles to the “real” verb while transparent regularization links the the verb to the “real” subjects or objects. The sentence below shows an example of subject control:

The army planned to reach the destination on time. (5.11)

The only difference shown below is that with subject control the word “reach” has a subject “army”. Then the new syntactic structure contains the meaning that “The army reaches, or is supposed to reach the destination”. Therefore this sentence is much more likely to be extracted as an event instance of type “Transportation”.

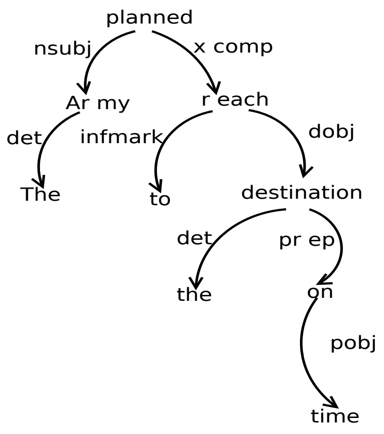


Figure 5.13: Original Dependency Tree without *Subject Control*

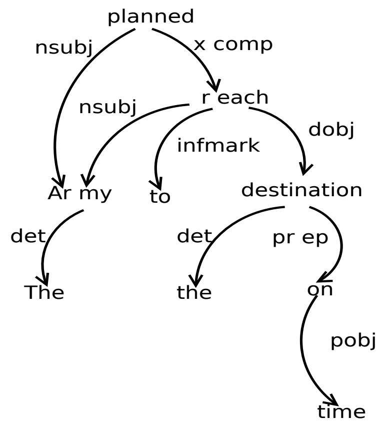


Figure 5.14: Dependency Tree with *Subject Control*

5.1.7 Subject Raising

In linguistics, raising is the construction where a given predicate/verb takes a dependent that is not its semantic argument, but rather it is the semantic argument of an embedded predicate. In other words, an argument that belongs to an embedded predicate is realized syntactically as a dependent of a higher predicate/verb. Subject Raising is the situation where the lower predicate selects the subject. For example,

1. John seemed to believe
2. John seemed to make sense
3. The idea seemed to make sense
4. The idea seemed to fall

Anything can be the subject of “seem”, but the lower predicate may place restrictions on the subject. The sentence 4 doesn’t make sense, even though it is syntactically well-formed – it might be found in poetry or in other weird writing. The influence of subject raising on syntactic structures is much clearer than subject control because the raising verb is almost useless in understanding the sentence since we just jump over that verb and link the subject with the “real” verb. The sentence below shows an example of subject raising:

The army seemed to reach the destination on time. (5.12)

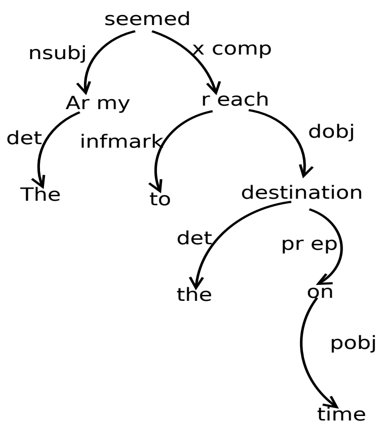


Figure 5.15: Original Dependency Tree without *Subject Raising*

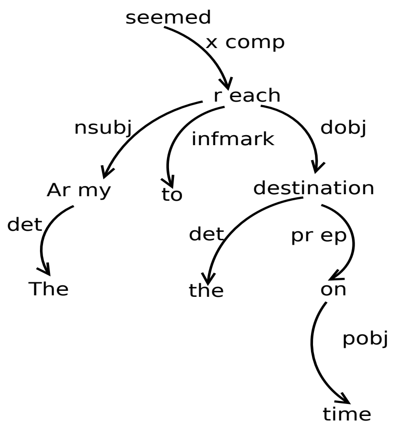


Figure 5.16: Dependency Tree with *Subject Raising*

In the instance above the army actually "reach" the place however from the original dependency tree we would never reach the "real" subject "army" from the word "reach" through dependency paths. However after raising control regularization, the dependency structure is much clearer, showing that in fact the army reached the destination, which is much more likely to be identified as a "Transportation" event.

5.2 Experiment

In this section, we will compare the performance of applying different dependency regularization methods with the event extraction baseline system, and discuss the contributions of these different dependency regularization rules.

Table 5.1 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.

Table 5.2 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 bell 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

Regularization	**Recall**	**Precision**	**F-score**
original	65.45	70.59	67.92
vch	66.82	70.84	68.77
pv	65.23	70.34	67.69
rc	65.45	70.59	67.92
sub	64.32	70.05	67.06
raising	65.45	70.42	67.84
transparent	65.68	71.18	68.32
vch & transp	67.27	71.50	69.32
vch & transp & Nomlex	68.18	72.82	70.42

Table 5.1: 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, *sub* – subject control regularization, *raising* – subject raising regularization, *transparent* – transparent regularization, *Nomlex* – nominalizatioin regularization

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 5.2: 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

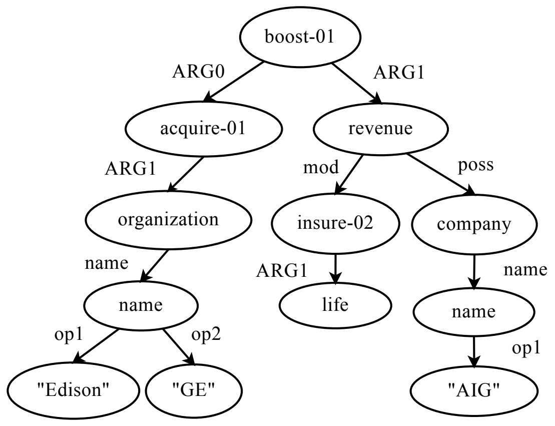
original framework.

Chapter 6

Improving Event Extraction with Abstract Meaning Representation 1

Abstract Meaning Representation (AMR) (Banarescu et al., 2013) is a sembanking language that captures whole sentence meanings in a rooted, directed, labeled, and (predominantly) acyclic graph structure - see Figure 6.1 for an example AMR parse. AMR utilizes multi-layer linguistic analysis such as PropBank frames, non-core semantic roles, coreference, named entity annotation, modality and negation to represent the semantic structure of a sentence. AMR strives for a more logical, less syntactic representation, collapsing some word category (verbs and nouns), word order, and morphological variation. Instead, it focuses on semantic relations between concepts and makes heavy use of predicate-argument

1. This chapter is mainly adapted from a published paper (Li et al., 2015)



(a) AMR graph

```
(b / boost-01
  :ARG0 (a / acquire-01
    :ARG1 (o / organization
      :name (n2 / name
        :op1 "Edison"
        :op2 "GE")))
  :ARG1 (r / revenue
    :mod (i / insure-02
      :ARG1 (l / life))
    :poss (c / company
      :name (n / name
        :op1 "AIG"))))
```

(b) AMR annotation

Figure 6.1: Two equivalent ways of representing the AMR parse for the example sentence, "*The acquisition of Edison GE will boost AIG's annual life insurance revenue.*"

structures as defined in PropBank (Kingsbury and Palmer, 2002; Palmer et al., 2005). For example, a phrase like "*bond investor*" is represented using the frame "`invest-01`", even though no verbs appear.

In addition, many function words (determiners, prepositions) are considered to be syntactic “sugar” and are not explicitly represented in AMR, except for the semantic relations they signal. Hence, it assigns the same AMR parse graph to sentences that have the same basic meaning.[2]

Compared to traditional dependency parsing and semantic role labeling, the nodes in AMR are entities instead of words, and the edge types are much more fine-grained. AMR thus captures deeper meaning compared with other representations which are more commonly used to represent context in event extraction. In this work, all AMR parse graphs are automatically generated from the first published AMR parser, *JAMR* (Flanigan et al., 2014).

6.1 Framework and Features

Feature	**Description**
amr_word_tag	Conjunction of the candidate word and its AMR tag
amr_dist_to_root	Distance between the candidate word and the root
amr_parent_word	Word of the parent node
amr_parent_tag	AMR tag of the parent node
amr_parent_word_tag	Conjunction of the parent word and its AMR tag
amr_sibling_tag	AMR tag of each sibling node
amr_sibling_word_tag	Conjunction of the sibling word and its AMR tag
amr_child_word_tag	Conjunction of the child word and its AMR tag
amr_grandchild_word	Word of the grandchild node

Table 6.1: Features extracted from the AMR graph

To compare our proposed AMR features with the previous approaches, we implemented a Maximum Entropy (MaxEnt) classifier with both traditional features

2. Readers can refer to Banarescu et al. (2013) for a complete description of AMR and more examples.

Node	**Feature**	**Example**
Candidate	amr_word_tag	`acquire-01_ARG0`
Root	amr_dist_to_root	`1`
Parent	amr_parent_word	`boost-01`
	amr_parent_tag	`AMR-Root`
	amr_parent_word_tag	`boost-01_AMR-Root`
Sibling	amr_sibling_tag	`ARG1`
	amr_sibling_word_tag	`revenue_ARG1`
Children	amr_child_word_tag	`org_ARG1`
Grandchildren	amr_grandchild_word	`name`

Table 6.2: Example features for candidate "*acquisition*".

and AMR features for trigger identification and label classification.

To make a fair comparison, the feature sets in the baseline are identical to the local text features in Li et al. (2013). From Table 6.3, we can see that this baseline MaxEnt classifier with local features aligns well with the joint beam search approach using perceptron and local features in Li et al. (2013). The slight variation is mainly due to the different pre-processing procedures for features.

On top of the local features used in the baseline MaxEnt classifier, we exploit knowledge from AMR parse graphs to add AMR features into the MaxEnt classifier. The effects of these features have been explored based on the performance on the development dataset. More features have actually been studied, such as the features extracted from the grandparent node, the conjunction features of candidate and parent nodes, etc. Table 6.1 lists the final AMR features extracted from the AMR parse graph. Table 6.2 lists the feature values for trigger candidate "*acquisition*", from the above example AMR graph.

6.2 Experiments

This section compares our MaxEnt classifiers using both baseline features and additional proposed AMR features with the state-of-the-art systems on the blind test set, and then discusses the results in more detail. The experiment is based on a feature-based system, therefore we compare the performance with an entirely different baseline from the other approaches.

Methods	P	R	F_1
Sentence-level in Hong et al. (2011)	67.6	53.5	59.7
MaxEnt classifier with local features in Li et al. (2013)	74.5	59.1	65.9
JBS with local features in Li et al. (2013)	73.7	59.3	65.7
JBS with local and global features in Li et al. (2013)	73.7	62.3	67.5
Cross-event in Liao and Grishman (2010) †	68.7	68.9	68.8
Cross-entity in Hong et al. (2011) †	72.9	64.3	68.3
MaxEnt classifier with baseline features	70.8	61.4	65.7
MaxEnt classifier with baseline + AMR features	74.4	62.3	**67.8**

Table 6.3: Performance (%) comparison with the state-of-the-art systems, where JBS – Joint Beam Search. † beyond sentence level.

Table 6.3 presents the overall performance of the systems with gold-standard entity mention and type information.

As we can see from Table 6.3, among the systems that only use sentence level information, our MaxEnt classifier using both baseline and AMR features significantly outperforms the MaxEnt classifier with baseline features as well as the joint beam search with local features from Li et al. (2013) (an absolute improvement of 2.1% in F_1 score), and performs comparably (67.8% in F_1) to the state-of-the-art joint beam search approach using both local and global features (67.5% in F_1) (Li et al., 2013). This is remarkable since our MaxEnt classifier does not require any global features[3] or sophisticated machine learning framework with a much larger hypothesis space, e.g., structured perceptron with beam search (Li et al., 2013).

Event Type	**Baseline**			**Baseline + AMR**		
	P	R	F_1	P	R	F_1
Transaction:Transfer-Ownership	50.0	11.1	18.2	62.5	18.5	28.6
Business:Start-Org	0.0	0.0	0.0	100.0	5.9	11.1
Justice:Trial-Hearing	80.0	80.0	80.0	83.3	100.0	90.9
Justice:Appeal	85.7	100.0	92.3	100.0	100.0	100.0
Conflict:Demonstrate	80.0	57.1	66.7	100.0	57.1	72.8
Justice:Arrest-Jail	75.0	50.0	60.0	83.3	83.3	83.3
Contact:Phone-Write	20.0	12.5	15.4	40.0	25.0	30.8
Personnel:Start-Position	80.0	33.3	47.1	66.7	33.3	44.4
Justice:Release-Parole	50.0	100.0	66.7	33.3	100.0	50.0
Contact:Meet	85.7	87.1	86.4	82.3	82.3	82.3

Table 6.4: Comparison between the performance (%) of baseline and AMR on a subset of event types.

From the detailed result analysis, we can see that the event trigger detection of most event types are significantly ($p < 0.05$) improved over the baseline setting. Many types gain substantially in both precision and recall, while only 4 out of 33 event types decrease slightly in performance. Table 6.4 presents the performance comparison for a subset of event types between the baseline and the classifier with both baseline and AMR features.

For instance, in the test sentence "*...have Scud missiles capable of **reaching** Israel ...*", the trigger candidate "*reach*" can be a *Conflict:Attack* event (as in this case) but also a *Contact:Phone-Write* event (e.g., "*they tried to reach their loved*

3. Global features are the features generated from several event trigger candidates, such as bigrams of trigger types which occur in the same sentence or the same clause, binary feature indicating whether synonyms in the same sentence have the same trigger label, context and dependency paths between two triggers conjuncted with their types, etc.

ones"). If the subject (`ARG0`) is a weapon (as in this example), it should be an *Attack* event. This pattern can be learned from a sentence such as "*The missiles …reach their target*". The AMR parser is able to look through "*capable of*" and recognizes that "*missiles*" is the subject (`:ARG0 m2/missile`) of "*reach*" in this example. Thus AMR features are able to help predict the correct event type in this case.

AMR can also analyze and learn from different forms of the same word. For example, there are two examples in the ACE corpus involving "*repay*", one using the verb ("*repaying*") and the other one using the noun ("*repayment*"), and both are classified as *Transaction:Transfer-money* event. AMR could learn from the "*repaying*" example about the correct event type and then precisely apply it to the "*repayment*" example.

The gains from adding AMR features show that the features and knowledge encoded in the AMR parse graphs can complement the information incorporated in the dependency parse trees and other traditional features.

6.3 Discussion

Applying the AMR features separately, we find that the features extracted from the sibling nodes are the best predictors of correctness, which indicates that the contexts of sibling nodes associated with the AMR tags can provide better evidence for word sense disambiguation of the trigger candidate as needed for event type classification. Features from the parent node and children nodes are also significant contributors.

Performance of the current AMR parser suffers from a lack of training data. For example,

1. *A tank* ***fired*** *on the Palestine Hotel.*

2. *The company* ***fired*** *its president.*

where two "*fired*" are assigned the same PropBank frame (a very coarse notion of word sense), "`fire-01`", rather than distinguishing the different senses here. As measured in the *JAMR* description paper (Flanigan et al., 2014), this parser only achieves 58% in F_1 on the test data using the full pipeline (concept identification and relation identification stages). An AMR parser trained on a larger corpus would help much more on this event extraction task and other Information Extraction tasks.

Chapter 7

Improving Event Extraction with Active Learning [1]

There has been growing interest over the last few years in applying active learning methods to reduce the annotation burden involved in developing corpus-trained NLP modules. Active learning has been applied to a variety of Information Extraction tasks, including name tagging, parsing, partial parsing, relation extraction, etc (Majidi and Crane, 2013). Fu and Grishman (2013) investigated active learning methods based on co-testing for training relation extractors for ACE relations. Liao and Grishman (2011) applied such methods for the active learning of ACE event extractors, although with a very different approach (based on the distribution of event triggers across sentences) from that proposed here.

1. This chapter is mainly adapted from a published paper (Cao et al., 2015a)

7.1 Pattern Expansion

Supervised training can be moderately effective in creating an Event Detection system, but the process of annotating the large corpus required for good performance can be very expensive and time-consuming. The ACE 2005 corpus, with about 300,000 words, is one of the largest such corpora, with detailed event annotations covering 33 event types. Nonetheless, many expressions of these event types are not included, limiting performance of the trained system.

To significantly improve coverage through supervised training would require annotation of a corpus several times larger, which would be prohibitively expensive. Instead we used an active learning approach, in which we identified common constructs which were not represented in the original training corpus, selected examples of these constructs from another unannotated corpus and presented these examples to the user for event annotation. The unannotated corpus we used is *EnglishGigaWord* [2], which is a comprehensive archive of newswire text data in English. The *EnglishGigaWord* corpus contains four distinct international sources of English newswire: Agence France Press English Service, The New York Times Newswire Service, The New York Times Newswire Service, The Xinhua News Agency English Service.

The process of selecting examples from *EnglishGigaWord* is described below in details:

1. Computing the frequency of dependency relations: Since our pattern-based framework is based on syntactic patterns taken from dependency parses, we

2. `https://catalog.ldc.upenn.edu/LDC2003T05`

select examples to be labeled based on their dependency relations. We use *EnglishGigaWord* to compute frequencies and select particular types of dependency relations (direct object and prepositional object).

2. Filtering Step: Select dependency relations for which the governor (verb) has appeared as a trigger in the training corpus but the dependency relation as a whole has not appeared in the training corpus.

3. For each high-frequency dependency relation, pick the sentence from *EnglishGigaWord* with at least 5 tokens whose dependency tree contains this dependency relation and maximizes the following ranking score function:

$$score(s) = \begin{cases} 0 & len(s) < 5 \\ \frac{\prod_{1 \le i \le n} freq(w_i)}{len(s)} & len(s) \ge 5 \end{cases} \tag{7.1}$$

where w_i is the ith word in the sentence s, $freq(w_i)$ is the frequency probability of word w_i in the corpus, and $len(s)$ is the number of tokens of the sentence s[3]. This metric favors short sentences with common words, which should be easy to label.

With this function, the most representative instance matching a pattern would be extracted. For example, if we try to find an instance containing the pattern "`take office`", the following sentence would be extracted:

He is to take office today.

3. The stop words are not counted here.

This sentence is an instance of the event *Start-Position*.

4. Add the selected sentences: Annotate the selected instances with respect to the presence of event triggers and incorporate the annotated instances into the training data set.

5. Compare the results: Compare the performance of event detection applying pattern expansion with the AceJet baseline (without pattern expansion)

7.2 Experiments

Methods	*P*	*R*	*F1*
Sentence-level in (Hong et al., 2011)	67.6	53.5	59.7
MaxEnt classifier with local features in (Li et al., 2013)	74.5	59.1	65.9
JBS with local features in (Li et al., 2013)	73.7	59.3	65.7
JBS with local and global features in (Li et al., 2013)	73.7	62.3	67.5
Cross-event in Liao and Grishman (2010) †	68.7	68.9	68.8
Cross-entity in (Hong et al., 2011) †	72.9	64.3	68.3
MaxEnt classifier with local features	70.8	61.4	65.7
AceJet baseline	70.6	65.5	67.9
AceJet system with active learning	72.0	68.9	**70.4**

Table 7.1: Performance (%) comparison with the state-of-the-art systems, where JBS – Joint Beam Search. † beyond sentence level.

Table 7.1 presents the overall performance of the systems with gold-standard entity mention and type information. We can see that our system with active learning can improve the performance over our baseline, and also advances the current state-of-the-art systems. In the test sentence, "*The president is to **take office** tomorrow*", for instance, the system with expanded patterns can correctly identify the *Personnel:Start-Position* event, whereas the AceJet baseline even failed to recognize it as an event instance. Another example is, "*... the anti-communist Gen. Suharto **seized power** in 1965*", where the expanded pattern successfully detects the event trigger with the correct type *Personnel:Start-Position*.

7.3 Performance & Discussion

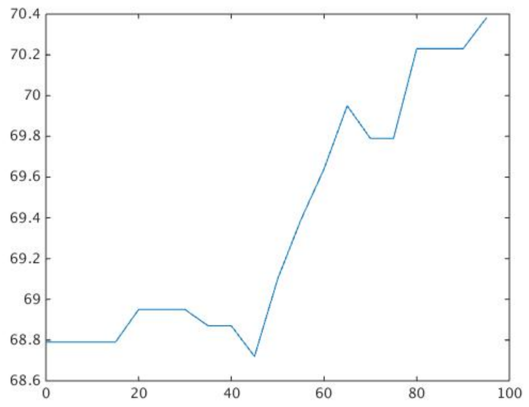


Figure 7.1: Semi-supervised pattern expansion performance (% in *F-Measure*)

In Figure 7.1, the x-axis is the number of instances added to the training data, while the y-axis is the corresponding F-measure. We can see from Figure 7.1 that the pattern expansion helps improve the performance (No substantial improvement has been found past 100 instances). However the improvement is only modest. This is mainly because the frequent dependency pairs may not be closely related to events and not all dependency pairs align with ACE event patterns very well. Since the pattern-based framework is based on matching dependency relation types and named entity types, noun groups play a central role to identify the events. Therefore, we focus on two types of frequent dependency relations:

- **direct object**

  The object of a verb plays a significant role in understanding the phrase. For example, the phrase "*take office*" means that a duty or title is assumed while other phrases like "*take an apple*" would not trigger an ACE event.

- **preposition and object**

  The noun in the prepositional phrase sometimes conveys as much or more information than the verb. For example, "*fight for independence*" is generally a *Demonstrate* event.

In contrast, there are three main classes of dependency relations which generally are not helpful in improving event extraction performance:

1. **Time Patterns**

   Time expressions generally do not help identify the event type. For example, the phrase "*tell Michael on Tuesday*' contains a time-modifying prepositional phrase "*on Tuesday*", but this time modifier plays little role in determining the type of the event. The verb "*tell*" is by itself a strong indicator of a *Contact* event, with the object also playing some role in the classification.

2. **Sports Patterns**

   Since ACE events are mainly about commercial and security-related news, patterns related to sports should be removed. For example, "*win a title*" is one of the top 5 high-frequency dependency pairs in the *EnglishGigaWord* corpus. This pattern appears mostly in a sports-related sentence or article.

3. **Redundant Patterns**

   Some verbs strongly favor a single event type. For example, "*die in hospital*" is a high-frequency pattern in *EnglishGigaWord*, however the verb "*die*" is sufficient to identify the *Die* event, whether a man dies in hospital, a room or on the road. Even if this pattern did not appear in the training data, adding it during pattern expansion will do little to improve event classifier accuracy because there are many *Die* events in the training data whose trigger is the verb "*die*". Other information from context will have minimal effect compared to the contribution of the verb "*die*" itself. We believe that such cases can be identified as patterns with triggers a large fraction of whose training examples represent the same event type.

Of the 100 examples tagged, 28 were positive (event triggers); of the 28, we considered 14 to be redundant (not helpful).

Chapter 8

Improving Event Extraction with Expert-level Patterns

This research combines TABARI and ACE event patterns. Therefore we need to start by introducing TABARI event patterns and some basic definitions. TABARI (Textual Analysis By Augmented Replacement Instructions)[1] is an open source pattern-based event extraction system. TABARI event patterns [2] are hand-maded patterns used to extract events in the TABARI system. These patterns cover international affairs, which is also a major topic of ACE events. Events in the TABARI system are divided into 20 event types and 324 event subtypes. The basic information of the twenty event types is shown in the table below.

From the event names we can see that the event definitions in TABARI are significantly different from ACE events. Therefore the incorporation of TABARI

1. `http://eventdata.parusanalytics.com/software.dir/tabari.html`
2. `http://brenocon.com/tabari_cameo_verbs.html`

Event types	*# of event subtypes*	*# of patterns*
Make public statement	10	1075
Appeal	25	574
Express intent to cooperate	26	1074
Consult	7	264
Engage in diplomatic cooperation	8	385
Engage in material cooperation	5	125
Provide aid	6	286
Yield	25	391
Investigate	5	129
Demand	25	179
Disapprove	11	543
Reject	26	462
Threaten	22	505
Protest	26	173
Exhibit military posture	5	191
Reduce relations	13	348
Coerce	12	317
Assault	13	85
Fight	7	431
Attack with weapons of mass destruction	7	5

Table 8.1: TABARI events

patterns with ACE patterns to perform ACE event extraction is not straightforward, but will require a combination of automatic processing and a limited amount of manual review.

8.1 Framework

Our goal in this chapter is to use the information from TABARI to generate additional ACE event patterns and thus improve the recall of our event extraction system. New patterns are generated in three steps, as shown in Figure 8.1.

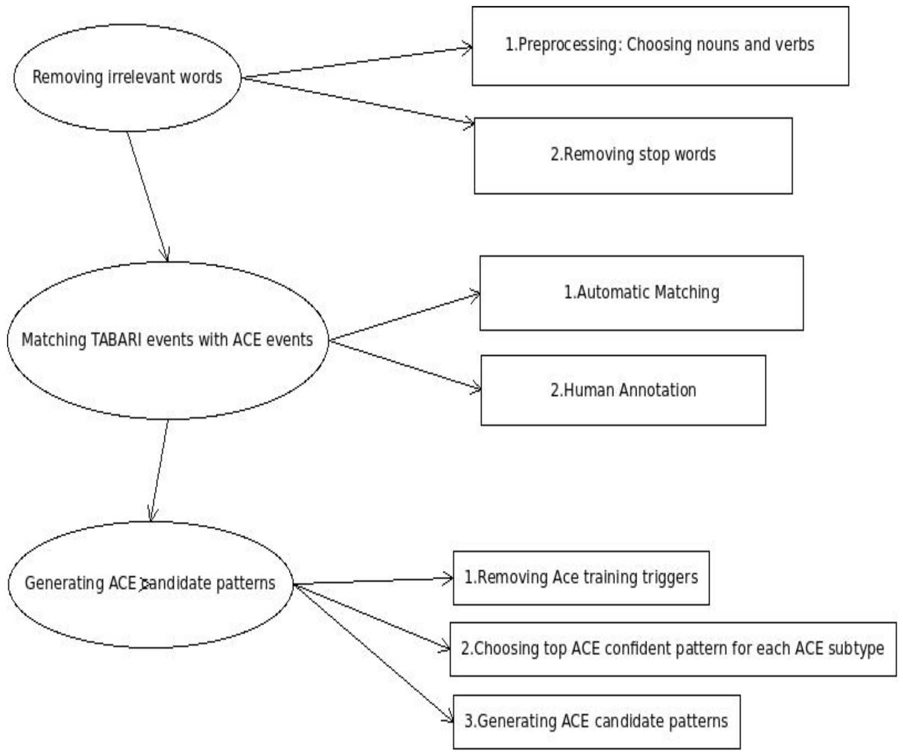

Removing irrelevant words
1.Preprocessing: Choosing nouns and verbs
2.Removing stop words
Matching TABARI events with ACE events
1.Automatic Matching
2.Human Annotation
Generating ACE candidate patterns
1.Removing Ace training triggers
2.Choosing top ACE confident pattern for each ACE subtype
3.Generating ACE candidate patterns

Figure 8.1: Generating ACE candidate patterns from TABARI event patterns

We start with the complete list of words appearing in TABARI patterns — a total of 2988 words. We eliminate most of these as likely ACE event triggers based on broad linguistic criteria: words other than nouns or verbs; argument nominalizations; nouns without complements; etc. The full set of criteria is presented below. This leaves 224 words which may be triggers.

Next we associate each candidate trigger with an ACE event subtype. This is the most difficult step as the type structure of ACE and TABARI events is so different.

1. **Automatic Subtype Alignment:** We compute the similarity between

ACE event subtypes and TABARI event subtypes, based on the overlap between the words appearing in TABARI patterns for the TABARI subtype and the ACE triggers for the ACE subtype. For each TABARI event subtype, we choose the most similar ACE event subtype. For example, the ACE event subtype "Conflict:Attack" is most similar to the TABARI event subtype "Threaten with military force, not specified below" (event subtype code : 138). We can see from the definition that an "Attack" is similar in meaning to "Threaten with military force".

2. **Choosing the corresponding ACE event subtype for each trigger candidate:** If the trigger candidate appears in the rules for only one TABARI subtype, we use the corresponding ACE subtype, as determined by subtype alignment. If the trigger candidate is associated with multiple TABARI subtypes, we resolve the ambiguity manually, choosing an ACE subtype from a list of ACE subtypes by hand. For example, the ACE candidate trigger "brief" appears in TABARI patterns of 4 TABARI event subtypes: 040(Consult), 173(Arrest), 020(Make an appeal or request) and 042(Make a visit). These four types' matching ACE event subtypes are "Contact:Meet", "Conflict: Attack", "Movement:Transport" and "Movement:Transport", respectively. We can see that the match between ACE and TABARI event subtypes is basically correct. The word "brief" appears in these four types of event patterns based on different contexts. Therefore we need to choose an ACE event subtype for the ACE candidate trigger "brief", which is "Contact:Meet".

   Out of the 224 candidate triggers, 49 words match only one ACE event subtype. Therefore we only have to annotate the other 175 candidate triggers with the proper matching ACE event subtype.

From the two steps above, we have a list of candidate triggers, and associated with each trigger an ACE event subtype. Finally we note that the pattern-based event extraction system must be based on event patterns. Therefore we have to generate new ACE event patterns for the new triggers. These patterns are generated as follows:

1. Removing triggers which appear in the ACE training corpus.
2. Choosing the most confident pattern for each ACE subtype from the ACE training patterns (the confidence is the event score from the baseline system, as described in Chapter 4).
3. Combining the trigger, the ACE subtype, and the high-confidence pattern and adding the resulting entry to the extraction system.

8.2 Reducing TABARI Pattern Words

Event triggers are mostly verbs and nouns. Most noun triggers are nominalizations. So we limit our consideration to verbs and nominalizations. Moreover, some verbs or nouns cannot be event triggers. For example, subject-controlling verbs are normally not event triggers, such as the verb “plan”, “want”, etc. For example, in the sentence “The army planned to attack the city”, an “*Attack*” event would occur. However the trigger is “attack” instead of “plan”. With the list of stop words including subject-controlling verbs, we can remove words that cannot become event triggers from the list of TABARI pattern words. Therefore the final list of ACE candidate triggers can be quite small and accurate.

The TABARI event patterns mainly focus on military and political information. However most of the words in the patterns are noise words instead of event triggers. To remove the noise words with as little human annotation as possible, lists of stop words are used to exclude TABARI candidate triggers. These lists comes from basic knowledge of linguistics and information extracted from the ACE corpus. Table 2 shows the size and examples of each word list.

1. **Argument-nominalizations:** Some nominalizations are more likely to be considered as event arguments than other nouns. These words are called argument-nominalizations. For example, an "attacker" is always an argument of event "Attack", but it cannot be an event trigger.

2. **Special Nouns:** This is the most speculative. Nouns that do not have any subcategorization are unlikely to be event signals. To get these find all the words in noun-list1 that are not in nomlex words. noun-list1 includes all words in COMLEX which do not have any subcategorization and nomlex words includes all the words in nomlex (which are thus assumed to have some arguments).

3. **Verbs with Small Clause Complements:** These are similar subject raising verbs. They basically connect a subject with a predicate, but also are not really actions.

For example, in “I like my meat well-done”, “like” connects “my meat” with “well-done”, meaning that I like situations in which “my meat is well-done”. So “like” does not really trigger any action. There are all different kinds of small clauses, but they all connect some noun phrases with some predicate.

4. **Subject Raising:** Subject Raising is the situation that the lower predicate selects the subject. For example,

   John seemed to leave. (8.1)

   The sentence above might be an “Movement” event but the word “seem” can never become the triggers. Words like “seem” are called subject raising words.

5. **Transparent Nouns:** Most of the transparent nouns are quantifiers. For example, in “thousands of people”, the word “thousands” is the quantifier of “the people”. Event triggers are mostly nouns and verbs. Most of the noun event triggers contain almost all the information to identify the events. For example, the noun “war” is probably a “Conflict:Attack” event describing a war. “Appointment” must contain the information of a meeting between two persons, normally celebrities. Therefore transparent nouns cannot be event triggers. We include the transparent nouns into the lists of stop words.

6. **Money:** It is obvious that the words related to money, such as dollars or euros, are sometimes event arguments, like the amount of a transaction. However they cannot be event triggers.

7. **Time:** Time expression is similar to money words. A time entity can be an event argument instead of event triggers.

8. **ACE Event Arguments:** Normally Ace Entities are not event triggers. For example, in the sentence "50 civilians were killed in the attack", the event trigger is the word "kill" while "civilians" are the event argument.

Stop Words	*Size*	*Examples*
Argument Nominalizations	1,383	advisor, betrayer, sailor
Special Nouns	15,660	chicken, clubhouse, goalkeeper
Verbs with Small Clause Complements	219	remember, worry, love
Subject Raising	14	be, seem, begin
Transparent Nouns	408	cup, million, team
Money	114	dollars, euros, tax, cash
Time	147	afternoon, Monday, January
ACE event arguments	2,408	rebels, missle, immigrants

Table 8.2: The size and examples of each word list

8.3 Experiments

Table 8.3 presents the overall performance of the systems with gold-standard entity mention and type information. We can see that our system with information from expert-level patterns can improve the performance over our baseline, and also advances the current state-of-the-art systems.

Methods	*P*	*R*	*F1*
Sentence-level in Hong et al. (2011)	67.6	53.5	59.7
MaxEnt classifier with local features in Li et al. (2013)	74.5	59.1	65.9
JBS with local features in Li et al. (2013)	73.7	59.3	65.7
JBS with local and global features in Li et al. (2013)	73.7	62.3	67.5
Cross-event in Liao and Grishman (2010) †	68.7	68.9	68.8
Cross-entity in Hong et al. (2011) †	72.9	64.3	68.3
MaxEnt classifier with local features	70.8	61.4	65.7
AceJet baseline	70.6	65.5	67.9
AceJet system with expert-level patterns (TABARI)	65.2	75.2	**69.8**

Table 8.3: Performance (%) comparison with the state-of-the-art systems, where JBS – Joint Beam Search. † beyond sentence level.

Chapter 9

Combining approaches

9.1 Performance

Different techniques have been applied to the pattern-based systems to improve event extraction. Chapter 5 applies dependency regularizations to the original syntactic structure to improve the event extraction performance. Dependency regularizations help improve the event extraction with basic knowledge of linguistics. Dependency regularizations generalize the dependency structure to help more patterns match, however it never includes data from outside the ACE corpus. Chapter 7 includes event patterns that never appear in the training data, to improve the performance of event extraction. The patterns are extracted from another corpus (*EnglishGigaWord*). However all triggers of the patterns included have appeared in the training data. Since the ACE corpus is not evenly distributed, some event triggers in the test data actually never appear in the training set. Chapter 8 introduces new triggers from patterns of another event extraction program (TABARI). The new event triggers have not appeared in the training data. Since these three techniques are not strongly related to each other, it is necessary to see what happens if we combine all these three techniques [1].

Methods	P	R	F
AceJet baseline	70.6	65.5	67.9
Dependency Regularization in (Cao et al., 2015b)	72.8	68.2	70.4
Active Learning in (Cao et al., 2015a)	72.0	68.9	70.4
Expert-level patterns in chapter 8	65.2	75.2	69.8
Dependency Regularization & Active Learning	74.4	68.0	71.0
Dependency Regularization & Expert-level Patterns	71.1	70.9	71.0
Active Learning & Expert-level Patterns	72.1	70.4	71.2
All 3 approaches	72.6	71.1	71.8

Table 9.1: Performance comparison (%) of the combination of different approaches

The combination of dependency regularization and active learning helps improve the event extraction system. The improvement is minimal because of the overlap in the pattern set from the other corpus and the new event patterns with dependency regularization. Expert-level patterns also improve the performance of the event extraction system with the other approaches. This is because with expert-level patterns, we include new event triggers into the system, which is independent of other approaches.

9.2 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

1. Chapter 6 (AMR) is not included because it is based on a totally different feature-based system.

investigation, the potential performance of event extraction with the AceJet system ranges from 62.6% to 81.8%, where the upper bound and lower bound correspond to different modifications on top of the original event patterns.

1. **Upper Bound:** The ACE corpus is not evenly distributed. We add the test data into the training data, the event extraction performance F-score is 81.8%. Therefore whatever information we include, either from linguistics background knowledge (Chapter 5) or other corpus (Chapter 7) and systems (Chapter 8), 81.8% is the upper bound we can reach.

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.

Chapter 10

Conclusion

To date, the use of supervised methods for creating event extractors has been limited by their poor performance even using large annotated training corpora. The ACE corpus is both small and not evenly distributed. In this thesis, we present methods for improving event extraction. The first step is to explore the effectiveness of introducing linguistic knowledge (Dependency Regularization) and systematic tools (Abstract Meaning Representation) to boost Event Extraction performance. Then, we report on the effectiveness of including limited human annotations (Active Learning) to improve Event Extraction. Finally, we introduce the performance of a combination of our baseline system and event patterns from another event program (TABARI program). This step contains expert-level patterns generated by other research groups. Because the information received is complicated and quite different from the original corpus (ACE), the process of including this information requires more complicated processing.

The experimental results have demonstrated the effectiveness of *Dependency*

Regularization, *Abstract Meaning Representation*, *Active Learning* and *Expert-level patterns*, which has helped our pattern-based system achieve 2-3% absolute F-measure improvement over the baseline. Each approach advances the state-of-the-art systems.

Section 9.2 has shown that even if we add the test data into the training data, the event extraction performance cannot reach higher than 81.8%. Therefore whatever information we include, either from linguistics background knowledge or other corpus and systems, 81.8% is the upper bound we can reach.

Bibliography

Banarescu, Laura, Bonial, Claire, Cai, Shu, Georgescu, Madalina, Griffitt, Kira, Hermjakob, Ulf, Knight, Kevin, Koehn, Philipp, Palmer, Martha, and Schneider, Nathan (2013). "Abstract Meaning Representation for Sembanking". In: *Proceedings of ACL 2013 Workshop on Linguistic Annotation and Interoperability with Discourse.*

Bronstein, Ofer, Dagan, Ido, Li, Qi, Ji, Heng, and Frank, Anette (2015). "Seed-based Event Trigger Labeling: How far can event descriptions get us?" In: *Proceddings of ACL.*

Cao, Kai, Li, Xiang, Fan, Miao, and Grishman, Ralph (2015a). "Improving Event Detection with Active Learning". In: *Proceedings of RANLP.*

Cao, Kai, Li, Xiang, and Grishman, Ralph (2015b). "Improving Event Detection with Dependency Regularization". In: *Proceedings of RANLP.*

— (2016). "Leveraging Dependency Regularization for Event Extraction". In: *Proceedings of FLAIRS.*

Feng, Xiaocheng, Huang, Lifu, Tang, Duyu, Qin, Bing, Ji, Heng, and Liu, Ting (2016). "A Language-Independent Neural Network for Event Detection". In: *Proceddings of ACL.*

Finkel, Jenny Rose, Grenager, Trond, and Manning, Christopher (2005). "Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling". In: *Proceedings of ACL*.

Flanigan, Jeffrey, Thomson, Sam, Carbonell, Jaime G., Dyer, Chris, and Smith, Noah A. (2014). "A Discriminative Graph-Based Parser for the Abstract Meaning Representation". In: *Proceedings of ACL*, pp. 1426–1436.

Fu, Lisheng and Grishman, Ralph (2013). "An Efficient Active Learning Framework for New Relation Types". In: *Proceedings of the Sixth International Joint Conference on Natural Language Processing*.

Grishman, Ralph, Westbrook, David, and Meyers, Adam (2005). "NYU's English ACE 2005 System Description". In: *Proceedings of the ACE 2005 Evaluation Workshop*.

Hong, Yu, Zhang, Jianfeng, Ma, Bin, Yao, Jian-Min, Zhou, Guodong, and Zhu, Qiaoming (2011). "Using Cross-Entity Inference to Improve Event Extraction." In: *Proceedings of ACL*, pp. 1127–1136.

Huang, Ruihong and Riloff, Ellen (2012). "Bootstrapped Training of Event Extraction Classifiers". In: *Proceedings of EACL*.

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

Kingsbury, Paul and Palmer, Martha (2002). "From TreeBank to PropBank". In: *Proceedings of LREC*.

Li, Qi, Ji, Heng, and Huang, Liang (2013). "Joint Event Extraction via Structured Prediction with Global Features". In: *Proceedings of ACL*.

Li, Xiang, Nguyen, Huu Thien, Cao, Kai, and Grishman, Ralph (2015). “Improving Event Detection with Abstract Meaning Representation”. In: *Proceedings of ACL-IJCNLP Workshop on Computing News Storylines (CNewS 2015)*.

Liao, Shasha and Grishman, Ralph (2011). “Using Prediction from Sentential Scope to Build a Pseudo Co-Testing Learner for Event Extraction”. In: *Proceedings of 5th International Joint Conference on Natural Language Processing*.

— (2010). “Using Document Level Cross-Event Inference to Improve Event Extraction.” In: *Proceedings of ACL*, pp. 789–797.

Liu, Ting (2009). “Bootstrapping Events and Relations from Text”. PhD thesis. State University of New York at Albany.

Majidi, Saeed and Crane, Gregory (2013). “Active Learning for Dependency Parsing by A Committee of Parsers”. In: *IWPT-2013*.

Maslennikov, Mstislav and Chua, Tat seng (2007). “A Multi-resolution Framework for Information Extraction from Free Text”. In: *Proceedings of ACL*.

Nguyen, Thien Huu and Grishman, Ralph (2015). “Event Detection and Domain Adaptation with Convolutional Neural Networks”. In: *Proceddings of ACL*.

— (2016). “Modeling Skip-Grams for Event Detection with Convolutional Neural Networks”. In: *Proceddings of EMNLP*.

Nguyen, Thien Huu, Cho, Kyunghyun, and Grishman, Ralph (2016). “Joint Event Extraction via Recurrent Neural Networks”. In: *Proceddings of NAACL*.

Palmer, Martha, Kingsbury, Paul, and Gildea, Daniel (2005). “The Proposition Bank: An Annotated Corpus of Semantic Roles”. In: *Computational Linguistics* 31.

Patwardhan, Siddharth and Riloff, Ellen (2007). “Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions”. In: *Proceedings of EMNLP*.

Riloff, Ellen (1996). “Automatically generating extraction patterns from untagged text”. In: *Proceedings of AAAI*.

Stevenson, Mark and Greenwood, Mark (2005). “A semantic approach to ie pattern induction”. In: *Proceedings of ACL*.

Sudo, Kiyoshi, Sekine, Satoshi, and Grishman, Ralph (2003). “An improved extraction pattern representation model for automatic ie pattern acquisition”. In: *Proceedings of ACL*.

Surdeanu, Mihai, Turmo, Jordi, and Ageno, Alicia (2006). “Counter-training in discovery of semantic pattern”. In: *Proceedings of EACL 2006 Workshop on Adaptive Text Extraction and Mining (ATEM 2006)*.

Tratz, Stephen and Hovy, Eduard (2011). “Fast, Effective, Non-Projective, Semantically-Enriched Parser”. In: *Proceedings of EMNLP*.

Yangarber, Roman (2003). “Counter-training in discovery of semantic pattern”. In: *Proceedings of ACL*.

Yangarber, Roman, Grishman, Ralph, Tapanainen, Pasi, and Huttunen, Silja (2000). “Automatic acquisition of domain knowledge for information extraction”. In: *Proceedings of Coling*.