# Acquiring Information from Wider Scope to Improve Event Extraction

by

Shasha Liao

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

May, 2012

---

Ralph Grishman

© Shasha Liao

All Rights Reserved, 2012

To Emma, the most incredible miracle in my life.

# ACKNOWLEDGEMENTS

This research was supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058 and via Department of Interior National Business Center (DoI/NBC) contract number D11PC20154. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, DoI/NBC, or the U.S. Government.

Foremost, I would like to thank my advisor, Ralph Grishman, for teaching me how to do research. I have always admired and tried to learn from Ralph's enthusiasm in pursuing all kinds of new research problems, finding connections between different areas, and forging better abstractions and ways of understanding results.

Next, I would like to thank other mentors. Thank you, Adam Meyers and Satoshi Sekine, for all your helpful advice.

I would also like to thank Heng Ji, who is now in City University of New York, with whom I have discussed about all kinds of novel research ideas, not limited to this thesis. I also thank my co-workers, Ang Sun, Bonan Min, Wei Xu. I truly

enjoyed and learned much from these enlightening conversations, and thank them all for the many wonderful discussions and for all their hard work.

I thank my parents, Zhengtang Liao and Yaning Zhang, who first introduced me to the world and its many exciting mysteries, for their fully understanding, love, and sacrifices to my education abroad. This thesis would not have been possible otherwise.

Last, but certainly not least, I thank my husband, Roger, for all his love, support, encouragement, patience and care.

# ABSTRACT

Event extraction is a particularly challenging type of information extraction (IE). Most current event extraction systems rely on local information at the phrase or sentence level. However, this local context may be insufficient to resolve ambiguities in identifying particular types of events; information from a wider scope can serve to resolve some of these ambiguities.

In this thesis, we first investigate how to extract supervised and unsupervised features to improve a supervised baseline system. Then, we present two additional tasks to show the benefit of wider scope features in semi-supervised learning (self training) and active learning (co-testing). Experiments show that using features from wider scope can not only aid a supervised local event extraction baseline system, but also help the semi-supervised or active learning approach.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Introduction

While 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. 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) is the technique for automatically extracting structured data from text documents. Generally speaking, there are three levels of information extraction: entity extraction identifies all the useful snippets in text, such as people, location, and organizations; relation extraction identifies all the binary relations between entities, and event extraction identifies multi-way relations among entities.

At present, the techniques developed for named entity recognition are quite mature (Zhang et al. 2004), and the performance already achieves real practical usability. For relation and event extraction, the performance is much lower, and there are still a lot of issues to investigate.

Event extraction aims to extract the critical information about each event (the agent, objects, date, location, etc.) and place this information in a set of templates

(data base). Most of the current systems focus on processing single documents and, except for coreference resolution, operate a sentence at a time (Grishman et al., 2005; Ahn, 2006; Agichtein and Gravano, 2000; Hardy et al., 2006).

However, sometimes the local information is not enough, or is hard to extract. An event can be described in so many different ways in text, and the local context information from an individual sentence may not suffice to extract the event information accurately and completely. Moreover, even humans might need information from elsewhere inside or outside a document to make confident decisions about the information to be extracted. Thus, incorporating information from wider scope is one important direction to investigate.

Still, there are a lot of issues to be explored. Wider scope includes sentence, discourse, document, or cross-document levels and what kind of features can be extracted is still being investigated. There are several recent studies using high-level information to aid local event extraction systems. Depending on where the features come from, we divided them into three categories:

**Consistency enforcement**: the "one sense per discourse" hypothesis (Yarowsky 1995) was used to enforce consistency in the testing data. Such studies include work from Yangarber and Jokipii (2005), Yangarber (2006), Yangarber et al. (2007), Mann (2007), Ji and Grishman (2008), and Gupta and Ji (2009). This hypothesis is quite simple, and no learning is needed.

**Supervised features:** The second category extracts supervised features from a tagged corpus, typically, the event extraction training data. Such studies include work from Finkel et al. (2005), Maslennikov and Chua (2007), Ji and Grishman

(2008), Gupta and Ji (2009) and Patwardhan and Riloff (2007, 2009). These features are especially useful on traditional evaluations, where the training and testing data are similar.

**Unsupervised features:** The third category extracts unsupervised features from distributions in large-scale untagged corpora. Such studies include Riloff (1996), Yangarber et al. (2000). These features are used when there is not much training data, or the training and testing data has different distribution.

In this thesis, we first investigate how to extract supervised and unsupervised features to improve a supervised baseline system. Then, we present two additional tasks to show the benefit of wider scope features in semi-supervised learning (self training) and active learning (co-testing).

The thesis is organized as follows:

Chapter 1 describes the task of event extraction, and specifically the ACE 2005 evaluation. Chapter 2 introduces NYU's event extraction baseline system, which is a supervised model.

Chapter 3 shows how to use document level cross-event features to improve event extraction. Document level cross-event information was collected and encoded into features that are used to build a set of document-level classifiers. It was aimed to solve the cases where local (sentence level) information is not enough and evidence from wider scope is needed.

Chapter 4 shows how to embed large-scale unsupervised topic features to improve event extraction. Unsupervised topic models (LDA) were incorporated to improve event extraction both on test data similar to training data, and on more

balanced collections. Compared to a supervised multi-label classifier, the unsupervised approach can achieve comparable, even better, results. Also, we do not limit our study to the corpus from the standard ACE evaluation, which is preselected, but also investigate its performance on a regular newswire corpus.

Chapter 5 presents a self-training process for event extraction. Event-centric entity-level Information Retrieval (IR) techniques were incorporated to provide topic-related document clusters. Experiments showed that bootstrapping on such a corpus performed better than that on a regular corpus. Global inference based on the properties of such clusters was then applied to achieve further improvement.

Chapter 6 presents a pseudo co-testing method for event extraction, which depends on one view from the original problem of event extraction, and another view from a coarser granularity task. Moreover, multiple selection criteria were applied to seek training examples that are informative, representative, and varied.

Chapter 7 concludes the thesis as a whole, pointing out some possible directions for future work.

# Chapter 1

# Task Description

This chapter describes prior research done in information extraction (IE), focusing on event extraction, which involves the results of other IE tasks, like named entity recognition, entity identification and entity co-reference.

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 is now a considerable literature on event extraction, and in particular on supervised and semi-supervised methods for constructing such systems for new tasks. Previous work has sought to compare event extraction tasks by measuring the complexity of the linguistic representation of the information to be extracted and analyzing the distribution of information in the document[1].

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.

## *1.1 Automatic Content Extraction (ACE) Evaluation*

ACE 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

[1] http://www.nist.gov/speech/tests/ace/

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.[2]"

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.

**Entity Detection and Recognition (EDR)** 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. Nested mentions are also captured. Annotators also review the entire document to group mentions of the same entity together (Coreference).

**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.

[2] http://www.nist.gov/speech/tests/ace/

**Event Detection and Recognition (VDR)**. In VDR, annotators identify and characterize eight types of events in which EDR entities participate. Targeted types include Life, Movement, Transaction, Business, Conflict, Contact, Personnel, and Justice. Annotators tag the textual mention or trigger for each event, and categorize it by type and subtype. They further identify event arguments (agent, object, source and target) and attributes (temporal, locative as well as others like instrument or purpose) according to a type-specific template.

## *1.2 ACE Terminology*

Event extraction depends on previous phases like name identification, entity mention classification and coreference. **Table 1.3** shows the results of this preprocessing. Note that entity mentions that share the same EntityID are coreferential and treated as the same object. We first present some ACE terminology to understand this task more easily:

**Entity:** an object or a set of objects in one of the semantic categories of interest, referred to in the document by one or more (coreferential) entity mentions

**Entity mention:** a span of text which refers to an entity. Entity mentions may be referenced in a text by their name, indicated by a common noun or noun phrase, or represented by a pronoun.

**Value:** A Value is a string that further characterizes the properties of some Entity or Event. ACE is only interested in certain types of possible Values. Specifically, ACE annotates NUMERIC, CONTACT-INFO, TIMEX2, JOB-TITLE, CRIME and SENTENCE Values.

**Timex:** a time expression including date, time of the day, season, year, etc.

**Timex mention:** a reference to a Timex (typically, a noun phrase or specialized time pattern)

**Event:** a specific occurrence involving participants. An event is something that happens, which can frequently be described as a change of state. The ACE event extraction task only tags a particular set of types of events.

**Event mention[3]:** 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.

**Event mention trigger:** the main word that most clearly expresses an event occurrence. An ACE event trigger is generally a verb or a noun.

**Event mention arguments (roles):** the entity mentions that are involved in an event mention, and their relation to the event. For example, event Attack might include participants like Attacker and Target, or attributes like Time_Within and Place. Arguments will be taggable only when they occur within the scope of the corresponding event, typically the same sentence.

[3] Note that we do not deal with event mention coreference in this thesis, so each event mention is treated as a separate event.

### *1.2.1 ACE Event Overview*

There are eight event types in ACE: Life, Movement, Transaction, Business, Conflict, Contact, Personnel and Justice. And each event type contains one or several subtypes (see **Table 1.1**).

| **Event Type** | **Event Subtype** |
|---|---|
| Life | *Be-Born, Marry, Divorce, Injure, Die* |
| Movement | *Transport* |
| Transaction | *Transfer-Ownership, Transfer-Money* |
| Business | *Start-Org, Merge-Org, Declare-Bankruptcy, End-Org* |
| Conflict | *Attack, Demonstrate* |
| Contact | *Meet, Phone-Write* |
| Personnel | *Start-Position, End-Position, Nominate, Elect* |
| Justice | *Arrest-Jail, Release-Parole, Trial-Hearing, Charge-Indict, Sue, Convict, Sentence, Fine, Execute, Extradite, Acquit, Appeal, Pardon* |

**Table 1.1 - Event types and subtypes defined in ACE 2005**

| | | | |
|---|---|---|---|
| Person | Place | Buyer | Seller |
| Beneficiary | Price | Artifact | Origin |
| Destination | Giver | Recipient | Money |
| Org | Agent | Victim | Instrument |
| Entity | Attacker | Target | Defendant |
| Adjudicator | Prosecutor | Plaintiff | Crime |
| Position | Sentence | Vehicle | Time-After |
| Time-Before | Time-At-Beginning | Time-At-End | Time-Starting |
| Time-Ending | Time-Holds | Time-Within | |

**Table 1.2 - 35 Argument roles defined by ACE 2005**

There are a total of 35 argument roles defined for the above events (see **Table 1.2**). Each event type will have its own set of potential participant arguments for the entities that occur within its scope, while there are some general roles like Time-Arg and Place-Arg. There are two different kinds of arguments:

**Event Participants:** Event participants are *Entities* that are involved in the event. For example, an *Attack* event might contain *Attacker-Arg*, *Target-Arg*, and *Instrument-Arg,* which represent the attacking agent, the target of the attack, and the instrument used in the attack.

**Event Attributes:** Event attributes are Values that are involved in the event. There are two kinds of attributes: one is event-specific attributes, which includes *Crime-Arg*, *Sentence-Arg*, and *Position-Arg*, which are for events Crime, Sentence, and Personnel respectively; the other is general event attributes, which can be applied to most (if not all) events, including *Place-Arg* and *Time-Arg*.

### *1.2.2 ACE Event Example*

Here is an example:

> *(Ex 1–1) Three murders occurred in France today, including the senseless slaying of Bob Cole and the assassination of Joe Westbrook. Bob was on his way home when he was attacked…*

In (Ex 1–1), we will have the following entity or Timex mentions, with co-reference information (see **Table 1.3**). There are three Die events, which share the same Place and Time roles, with different Victim roles. And there is one Attack event sharing the same Place and Time roles with the Die events (see **Table 1.4** ).

| Entity mention | Head word | Entity ID | Entity type |
|---|---|---|---|
| 0001-1-1 | France | 0001-1 | GPE |
| 0001-T1-1 | Today | 0001-T1 | Timex |
| 0001-2-1 | Bob Cole | 0001-2 | PER |
| 0001-3-1 | Joe Westbrook | 0001-3 | PER |
| 0001-2-2 | Bob | 0001-2 | PER |
| 0001-2-3 | He | 0001-2 | PER |

**Table 1.3 – Entity and entity mentions and their types for** (Ex 1–1**)**

| Event type | Event subtype | Trigger | Role | | |
|---|---|---|---|---|---|
| | | | **Place** | **Victim** | **Time** |
| Life | Die | murder | 0001-1-1 | | 0001-T1-1 |
| Life | Die | death | 0001-1-1 | 0001-2-1 | 0001-T1-1 |
| Life | Die | killing | 0001-1-1 | 0001-3-1 | 0001-T1-1 |
| | | | **Place** | **Target** | **Time** |
| Conflict | Attack | attack | 0001-1-1 | 0001-2-3 | 0001-T1-1 |

**Table 1.4 – Event mentions and their types for** (Ex 1–1**)**

### *1.2.3 Evaluation Metric*

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.

We use the precision, recall, and F-measure standard metrics to evaluate the system performance, which are defined as follows:

$$\mathrm{Pr}ecision = \frac{|system\ \ samples \cap key\ \ samples|}{|system\ \ samples|}$$

$$\mathrm{Re}call = \frac{|system\ \ samples \cap key\ \ samples|}{|key\ \ samples|}$$

$$F - measure = \frac{2 * \mathrm{Pr}ecision * \mathrm{Re}call}{\mathrm{Pr}ecision + \mathrm{Re}call}$$

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<br>Trigger start offset<br>Trigger end offset |
| Argument Identification | Event type and subtype<br>Argument head start offset<br>Argument head end offset |
| Argument Classification | Event type and subtype<br>Argument head start offset<br>Argument end offset<br>Argument role |

**Table 1.5 - The elements that need to be matched for each evaluation metric**

# Chapter 2

# Baseline System

In this section, we describe a state-of-the-art English baseline event extraction system (Grishman et al. 2005), and possible problems in such systems. Our English extraction system, developed over the course of the last several ACE evaluations, includes facilities for the EDR (entity), RDR (relation), and VDR (event) tasks. The English ACE system is built on top of the JET (Java Extraction Toolkit), which was developed at NYU for instructional purposes and is freely available for research purposes.

This chapter is organized as follows: section 2.1 describes the pre-processing phases of event extraction; section 2.2 gives details on how to extract ACE events; sections 2.3 and 2.4 discuss the problems of trigger identification and argument identification in this baseline system.

## *2.1 Pre-processing Phases*

Before event extraction, there are some normal pre-processing steps. For example, we need to tokenize the words, and tag the part-of-speech. Also, since the arguments of the event should be entities, timex expressions, or values, we need to extract them first. We list the most important pre-processing steps below:

**Lexical Lookup:** The input text is divided into sentences and tokenized. Tokens are looked up in a large general English dictionary that provides part-of-speech information and the base form of inflected words.

**Named Entity Analysis:** Named entities are tagged using an HMM trained on the ACE training corpora. The HMM has six states for each name type (PERSON, GPE etc.), as well as a not-a-name state. These six states correspond to the token preceding the name; the single name token (for names with only one token); the first token of the name; an internal token of the name (neither first nor last); the last token of the name; and the token following the name. These multiple states allow the HMM to capture context information and limited information about the internal structure of the name.

**Reference resolution:** Reference resolution first identifies mentions (referring expressions) and then links co-referring expressions. Coreference resolution proceeds incrementally. Mentions are resolved in order of appearance in the document, and feature extraction depends on previous decisions. For each mention, the system checks to see whether a simple, high-precision rule can resolve the mention (whether there is an exact string match between two multi-word names, for example), and immediately applies the rule if one is found.

**ACE Entity Detection and Recognition:** Given the output of reference resolution, ACE entity detection is primarily a task of semantic classification of the co-referring mention groups. Classification is performed differently for entities with and without named mentions. If there is a named mention, the type of the entity is obtained from the name tagger. The subtype is determined using a

MaxEnt model whose features are the individual tokens of the name, trained on the ACE 2005 corpus. The type and subtype of a nominal mention is determined from the head of the mention (with the exception of a small number of hand-coded cases), with the frequencies of the types and subtypes learned from the ACE corpus.

## *2.2 Event Extraction Phase*

The identification of the trigger and the arguments interact: the relation between the trigger and the argument is one essential factor to identify both the trigger and the role of the argument. For example, if we know that the object of the word "shoot" is a person and it has the "fire a shot" sense, we can confidently identify the person as the *Target* role, and tag "shoot" as the trigger of an *Attack* event. As a result, most current event extraction systems consider trigger and argument information together to tag a reportable event.

Our baseline system is trigger-based, and combined pattern matching and statistical models. In training, there are three steps:

**Step 1:** For each instance of an event, we construct the chunk patterns representing the connection between the trigger and the event arguments. The chunk-based pattern records the sequence of constituent heads separating the trigger and arguments. For each argument, we record its ACE type and subtype as well as its head.

**Step 2:** For each pattern, we determine the accuracy of predicting an event given a partial match to it. We record the fraction of matches to the trigger alone

that predicts a correct event, and the fraction of matches to the trigger and each argument as well. Patterns that over-generate – predicting incorrect or spurious events most of the time – are discarded.

**Step 3:** Patterns are supplemented by a set of Maximum Entropy statistical classifiers:

**Argument Classifier:** it is a binary classifier that distinguishes arguments of a potential trigger from non-arguments.

**Role Classifier:** it is a multi-class classifier to assign each argument with its proper role.

**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.

| Classifier | Features |
|---|---|
| Argument/role classifier | • Trigger word<br>• Event type<br>• Argument mention type<br>• Head of mention<br>• Head of mention coupled with event subtype<br>• Word preceding mention<br>• Chunk path<br>• Chunk path coupled with event type<br>• Distance to trigger<br>• Syntactic path |
| Trigger classifier | • Trigger word |

| | |
|---|---|
| | • Fraction of times the trigger is reportable<br>• Probability that it has each argument |

**Table 2.1 - Features used in classifiers of the baseline system**

In the test procedure, each document is scanned for instances of triggers from the training corpus. When an instance is found, the system tries to match the environment of the trigger against the set of patterns associated with that trigger. This pattern-matching process, if successful, will assign some of the mentions in the sentence as arguments of a potential event mention. The argument classifier is applied to the remaining mentions in the sentence; for any argument passing that classifier, the role classifier is used to assign a role to it. Finally, once all arguments have been assigned, the reportable-event classifier is applied to the potential event mention; if the result is successful, this event mention is reported.[4]

## *2.3 Problems of Trigger Identification*

Identifying the trigger – the word most clearly expressing the event - is essential for event extraction. Usually, the trigger itself is the most important clue in detecting and classifying the type of an event. For example, the word "attack" is very likely to represent an *Attack* event while the word "meet" is not. However, this is not always the case. When we collect all the words that serve as an Attack event trigger at least once, and plot their probability of triggering an event (**Figure**

[4] If the event arguments include some assigned by the pattern-matching process, the event mention is accepted unconditionally, bypassing the reportable- event classifier.

**2.1**), we see that the probabilities are widely scattered. Some words always trigger an event (probability = 1.0), but most are not.

Why is the trigger itself not sufficient to identify an event? If we look at all the possible trigger words in ACE 2005, we find that a word can sometimes be a trigger, sometime not. For example, "attack with a stone" is an Attack event, since it might cause physical damage, while "verbal attack" is not.

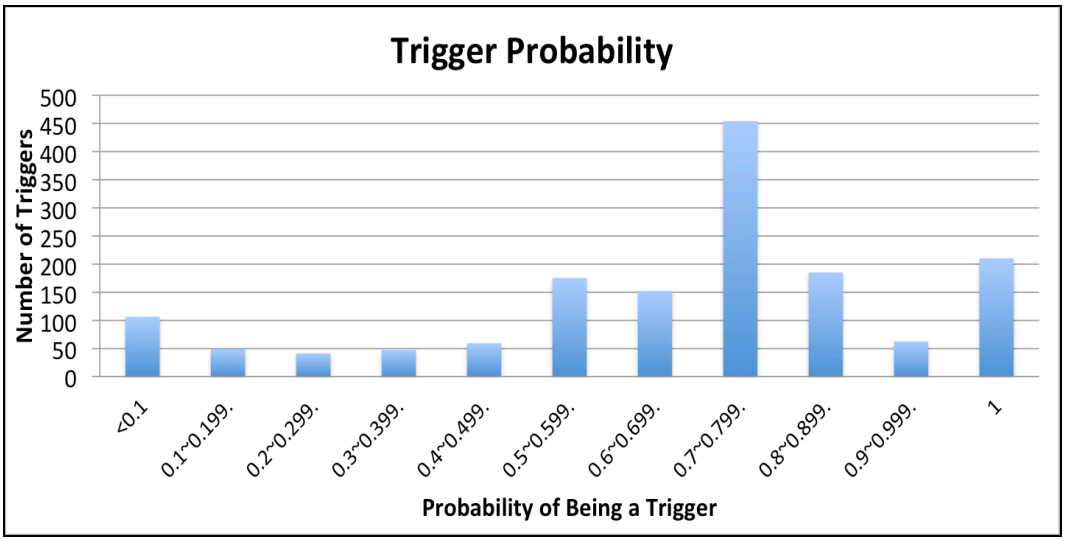


**Figure 2.1 - Distribution of probability of a word being a trigger for an Attack event**

| **Word** | **Event subtype** |
|---|---|
| casualties | *Die, Injury* |
| named | *Start-Position, Nominate, Elect* |
| pay | *Fine, Transfer-Money, Transfer-Ownership* |
| take | *Transport, Transfer-Ownership, Transfer-Money, Elect* |
| fire | *End-Position, Attack* |
| leave | *End-Position, Transport* |
| become | *Elect, Start-Position* |
| replace | *Start-Position, End-Position* |

| | |
|---|---|
| shot | *Attack, Die, Execute* |
| deliver | *Transport, Be-Born, Transfer-Ownership, Phone-Write* |

**Table 2.2 - Examples of ambiguous triggers**

Also, a word might trigger different types of events. To give some intuition, Table 2.2 presents several words that might trigger different types of events in different contexts. Below, we list some problems in trigger identification.

### *2.3.1 Word Sense Ambiguity*

A word may be ambiguous and have several senses, only some of which correspond to a particular event type. For example, the word "fire" can trigger Attack or End-Position events, based on different senses it presents in different contexts. If it means "the act of firing weapons or artillery at an enemy", it always triggers an Attack event; if it means "discharge from an office or position", it always triggers an End-Position event. Also, the sense "fire a shot", is more likely to trigger an Attack event, than the sense "record on photographic film".

### *2.3.2 Argument Constraint*

Even if the scenario is well detected, there is no guarantee of identifying the event correctly. Think about words like "fire" or "shot": these can only be an Attack event when the target is a person, organization, Geo-Political Entity (GPE), weapon or facility. If the target is, for example, an animal, it is probably not an Attack event.

### *2.3.3 Scenario (Context) Constraint*

The scenario also has subtle effects even once the correct sense is identified. For example, if we see the word "fire" with the sense "fire a shot" in terrorist activities, it may be an Attack event. However, the same word with the same sense in hunting-related or shooting-contest-related activities is probably not an Attack event.

However, we have to point out that while the above information can solve the problem to some extent, there are always some harder cases, which even humans cannot annotate confidently. For example, "casualties" can trigger both Die or Injury events, and most of the time, it is mentioned briefly after the report of an Attack event, and even a human reader cannot know whether it refers to death or injury, or sometimes, both.

## *2.4 Problems of Argument Identification*

Argument identification has its own problems as well, even if the trigger identification is correct. If we look at the performance of argument identification, we find that it is relatively low. **Figure 2.2** shows that the precision using the original training data is not very good: while precision improves with increasing classifier threshold, about 1/3 of the roles are still incorrectly tagged at a threshold of 0.90.

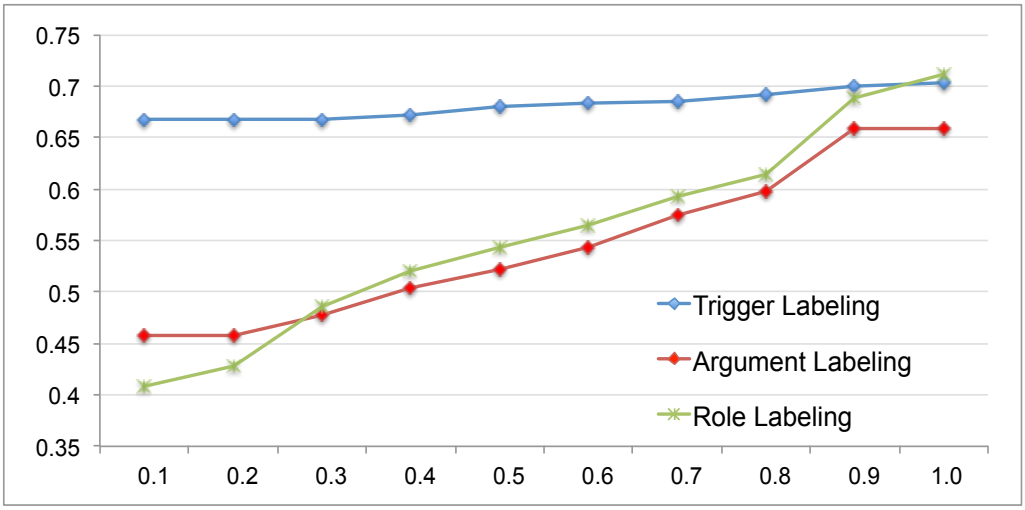


**Figure 2.2 – Average precision (5-folded cross-validation) of the baseline system with different confident thresholds**

There are several reasons for this:

### *2.4.1 Preprocessing Error*

Argument identification depends heavily on pre-processing such as EDR (Entity Detection and Recognition). Arguments are limited to Entities, Values and Timex which are found in previous phases. If the correct argument is not discovered, it will be never be identified by the event extraction system. Also, if its type and subtype are wrongly tagged, the event extraction will also be affected because they are important features for argument /role classification. For example:

*(Ex 2–1) Over 45 minutes as we watched, 32 patients were delivered here, evaluated and nine operations started; [29: **Artifact**] of those delivered [**Transport**], Marines involved in a firefight in the center of Baghdad.*

*(Ex 2–2) In the case of 1991, the task was to go in and get them out of Kuwait, and they did it, and [they: **Artifact**] were properly greeted coming back [**Transport**] to the United States.*

In Ex 2–1**Error! Reference source not found.**, the "29" should be tagged as a PERSON entity in ACE, however, the pre-processing misses it because the word itself is a number, and you need a inference from its context to decide it refers to "29 patients", which is very hard. Since the preprocessing misses this entity, we will never be able to tag it as the *Artifact* role for the *Transport* event.

In Ex 2-2, the preprocessing wrongly tags "they" as a "GPE" because the most recent entity mention for this pronoun is "Kuwait", which is a GPE entity. Since the *Artifact* role of *Transport* must be a person, the event tagger will not consider it as an *Artifact* role. Preprocessing misses 24% of the arguments, and 5.7% of the arguments' Entity/Values/Timex types are wrongly tagged, while 9.3% of the subtypes are wrongly tagged. Thus, the event extraction system would probably miss or wrongly assign roles for these mentions.

### *2.4.2 Structure Variation*

There are various ways to present an event, and the arguments can appear in many different positions. ACE evaluation already limits the argument identification to local context: the sentence, but that is not enough. Most event extraction systems identify the argument of a specific event by its relation to the trigger, for example, the chunk-based or syntax-based path from argument to trigger. This feature is a very strong predictor to identify the argument; for example, the subject of "kill" should fill the Agent role of a Die event, while the object should fill the Victim role. However, one big problem with this pattern is that people might use very different expressions and the path varies a lot. From **Figure 2.3**, we can see that most chunk-based paths appear only once or twice in

ACE data, which means that we cannot depend on such features since their frequency of occurrence is very low. Moreover, even if we use deeper structure, like semantic labeling, we do not come close to solving this problem.

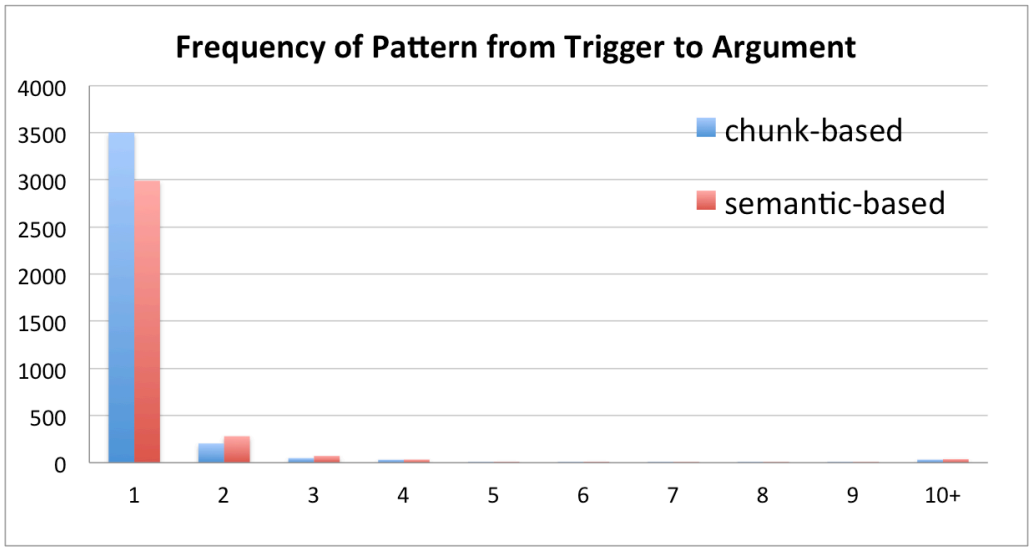


**Figure 2.3 - Frequency of patterns from trigger to argument**[5]

Also, there are arguments that are far from the trigger, for example, in a different clause of the sentence. In such a case, a little difference in the sentence might change the whole situation. Here is an example:

> *(Ex 2–3) He gave us a whole lot of information and then went [home:* ***Place****] and his [father-in-law:****Agent****]* killed*[****Die****]* *[him:****Victim****].*
>
> *(Ex 2–4) He gave us a whole lot of information and then left home and his [father-in-law:****Agent****] killed[****Die****] [him:****Victim****].*

[5] Note that the argument is replaced by its ACE type (GPE, Person, Organization, etc.), and the trigger is replaced by its event type (Attack, Die, Start-Position, etc.). X axis is the pattern occurrence frequency, while Y axis is the number of patterns with that frequency

In the above examples, a *Die* event occurs in each sentence, and there is only one word different for the two sentences. The annotation for trigger identification is easy: "kill" is a very strong prediction word for event *Die*. Also, the Victim argument is easy to detect since it is the object of the verb kill. However, the *Place* argument is hard to detect because it is in another clause, and a different expression might cause different results. In (Ex 2–3), a human can easily tag "home" as the *Place* argument of the *Die* event, while in (Ex 2–4), a human would probably not tag it as an argument of the *Die* event.

### *2.4.3 Multi-mention Role Problem*

Last but still, not less important, ACE argument identification allows one event to have more than one mention fill the same role. 16.6% of arguments share the same role with other arguments. Also, the "multiple mention in one slot" problem is not limited to a few roles; it appears in 28 roles (see **Table 2.3**) out of a total of 35 roles. Our baseline system uses a "one slot one mention" policy, and assigns the mention with highest probability to a specific slot. We did this because we haven't found an efficient way to deal with the multi-mention role problem. Here is an example:

> *(Ex 2–5) Clinton suffered greatly over the ［19 Rangers:**Victim**] that died, [18:**Victim**] on the 3rd of October and [Matt Reersen:**Victim**] three days later.*

The baseline system can easily find "Rangers" as victim, but it will miss the "18" and "Matt Reersen".

| | | | |
|---|---|---|---|
| Entity | Money | Destination | Adjudicator |
| Time-Within | Recipient | Artifact | Buyer |
| Victim | Target | Crime | Prosecutor |
| Position | Attacker | Plaintiff | Vehicle |
| Person | Place | Sentence | Time-Holds |
| Giver | Instrument | Agent | Price |
| Org | Origin | Defendant | Beneficiary |

**Table 2.3 - Argument roles that can contain multiple mentions**

## *2.5 How to Solve the Above Problems*

Although the problems are clear from the above sections, how to solve them is still under investigation. The local information can solve some of them to some extent. For example, in the baseline system (see section 2.2), the argument mention type is used to solve the argument constraint (see section 2.3.2). The fraction of times when the trigger is reportable is used to figure out whether this trigger is too ambiguous or not, which can partially solve the word sense ambiguity problem (see section 2.3.1). The distance from argument to trigger can be used to generalize over different structure variations (see section 2.4.2), although it might be too general. However, to thoroughly solve all the problems might be hard or even impossible: Word sense disambiguation is already a hard NLP task, and normalizing different expressions is even harder. Moreover, when the training

data is preselected, these problems would be more severe (see details in section 4.1).

Therefore, we need to figure out some indirect but novel ways to solve the problems. That is, how do we extract information from non-local scope to aid the baseline system, without too much effort? In chapter 3, we will discuss how to solve the above problems without really doing word sense disambiguation, structure normalization, or context detection, but using some indirect information instead. In chapter 4, we will investigate how to solve the problem when the data is imbalanced.

At last, we have to point out we do not investigate the problem of preprocessing error, and the multi-mention role problem in this thesis, since we do not have a clear idea of how features from wider scope can help improve them. We just point them out to explain why the baseline system has relatively low performance.

# Chapter 3

# Document Level Cross-event Inference[6]

Most current event extraction systems are based on phrase or sentence level extraction. This local context may be insufficient to resolve ambiguities for both trigger and argument identification; sometimes it is difficult even for people to classify events from isolated sentences. However, information from a wider scope

[6] This chapter is mainly adapted from a published paper (Liao and Grishman 2010a).

can serve to resolve some of these problems to some extent. For trigger identification, wider scope, instead of the local sentence, might be a better predictor for both word sense disambiguation or scenario detection (see section 2.3). For example:

*(Ex 3–1) He **left** the company.*

It is hard to tell whether it is a *Transport* event in ACE, which means that he left the place; or an *End-Position* event, which means that he retired from the company.

However, if there are similar events reported in the document, they will be good predictors. For example, a sentence "*he planned to go shopping before he went home*" would give us confidence to tag it as a *Transport* event; while another sentence like "*They held a party for his retirement*" would lead us to tag it as an *End-Position* event.

Moreover, different events are also a good predictor for a specific event. For example, if we find a *Start-Position* event like "*he was named president three years ago*", we are also confident to tag (Ex 3–1) as an *End-Position* event.

Event argument identification also shares this benefit, especially when the sentence structure is complicated or ambiguous (see section 2.4.2). In such cases, obtaining more information from the sentence structure might be hard, but if we just check the co-occurrence of other events in the document, we can acquire evidence in a much easier way. Consider the following two sentences:

*(Ex 3–2) A bomb **exploded** in Bagdad; seven people **died** while 11 were **injured**.*

*(Ex 3–3) A bomb **exploded** in Bagdad; the suspect got **caught** when he tried to escape.*

If we only consider the local context of the trigger "*exploded*", it is hard to determine that "*seven people*" is a likely *Target* of the *Attack* event in (Ex 3–2), or that the "*suspect*" is the *Attacker* of the *Attack* event, because the structures of (Ex 3–2) and (Ex 3–3) are quite similar. The only clue is from the semantic inference that a person who died may well have been a *Target* of the *Attack* event, and the person arrested is probably the *Attacker* of the *Attack* event. These may be seen as examples of a broader textual inference problem, and in general such knowledge is quite difficult to acquire and apply. However, in the present case we can take advantage of event extraction to learn these rules in a simpler fashion, which we present below.

As mentioned in the Introduction, there are several studies on extracting features on wider scope. Maslennikov and Chua (2007) used discourse trees and local syntactic dependencies in a pattern-based framework to incorporate wider context to refine the performance of relation extraction. They claimed that discourse information could filter noisy dependency paths as well as increasing the reliability of dependency path extraction.

Finkel et al. (2005) used Gibbs sampling, a simple Monte Carlo method used to perform approximate inference in factored probabilistic models. By using simulated annealing in place of Viterbi decoding in sequence models such as HMMs, CMMs, and CRFs, it is possible to incorporate non-local structure while preserving tractable inference. They used this technique to augment an information

extraction system with long-distance dependency models, enforcing label consistency and extraction template consistency constraints.

Ji and Grishman (2008) were inspired from the hypothesis of "One Sense Per Discourse" (Yarowsky, 1995); they extended the scope from a single document to a cluster of topic-related documents and employed a rule-based approach to propagate consistent trigger classification and event arguments across sentences and documents. Combining global evidence from related documents with local decisions, they obtained an appreciable improvement in both event and event argument identification.

Patwardhan and Riloff (2009) proposed an event extraction model which consists of two components: a model for sentential event recognition, which offers a probabilistic assessment of whether a sentence is discussing a domain-relevant event; and a model for recognizing plausible role fillers, which identifies phrases as role fillers based upon the assumption that the surrounding context is discussing a relevant event. This unified probabilistic model allows the two components to jointly make decisions based upon both the local evidence surrounding each phrase and the "peripheral vision".

Gupta and Ji (2009) used cross-event information within ACE extraction, but only for recovering implicit time information for events.

However, most of the above work focuses on single event extraction, or focuses on high-level information within a single event type (or a scenario), and does not consider information acquired from other event types. We extend these approaches

by introducing cross-event information to enhance the performance of multi-event-type extraction systems.

This chapter is organized as follows: section 3.1 will introduces the motivation of using cross-event information, and the consistency and distribution for trigger and argument respectively; section 3.2 will demonstrate how the cross-event information is extracted and encoded into the baseline system; section 3.3 shows the improvement we obtained in experiments.

## *3.1 Motivation*

Cross-event information is quite useful: first, some events co-occur frequently, while other events do not. For example, *Attack, Die*, and *Injure* events very frequently occur together, while *Attack* and *Marry* are less likely to co-occur. Also, typical relations among the arguments of different types of events can be helpful in predicting information to be extracted. For example, the Victim of a Die event is probably the Target of the Attack event. As a result, we extend the observation that "a document containing a certain event is likely to contain more events of the same type", and base our approach on the idea that "a document containing a certain type of event is likely to contain instances of related events".

We analyzed the sentence-level baseline event extraction, and found that many events are missing or spuriously tagged because the local information is not sufficient to make a confident decision. In some local contexts, it is easy to identify an event; in others, it is hard to do so. Thus, if we first tag the easier cases, and use such knowledge to help tag the harder cases, we might get better overall

performance. In addition, global information can make the event tagging more consistent at the document level.

Here are some examples. For trigger classification:

*(Ex 3–4) The pro-reform director of Iran's biggest-selling daily newspaper and official organ of Tehran's municipality has **stepped** down following the **appointment** of a conservative …it was **founded** a decade ago … but a conservative city council was **elected** in the February 28 municipal polls … Mahmud Ahmadi-Nejad, reported to be a hardliner among conservatives, was **appointed** mayor on Saturday …**Founded** by former mayor Gholamhossein Karbaschi, Hamshahri…*

The sentence level baseline system finds event triggers like "*founded*" (trigger of *Start-Org*), "*elected*" (trigger of *Elect*), and "*appointment*" (trigger of *Start-Position*), which are easier to identify because these triggers have more specific meanings. However, it does not recognize the trigger "*stepped*" (trigger of *End-Position*) because in the training corpus "*stepped*" does not always appear as an *End-Position* event, and local context does not provide enough information for the MaxEnt model to tag it as a trigger. However, in the document that contains related events like *Start-Position*, "*stepped*" is more likely to be tagged as an *End-Position* event.

For argument classification, the cross-event evidence from the document level is also useful:

*(Ex 3–5) British officials say they believe **Hassan** was a blindfolded woman seen being **shot** in the head by a hooded militant on a video obtained but not aired by the Arab television station Al-Jazeera. **She** would be the first*

*foreign woman to **die** in the wave of kidnappings in Iraq…**she**'s been **killed** by (men in pajamas), turn Iraq upside down and find them.*

From this document, the local information is not enough for our system to tag "*Hassan*" as the target of an *Attack* event, because it is quite far from the trigger "*shot*" and the syntax is somewhat complex. However, it is easy to tag "*she*" as the *Victim* of a *Die* event, because it is the object of the trigger "*killed*". As "*she*" and "*Hassan*" are co-referential, we can use this easily tagged argument to help identify the harder one.

### *3.1.1 Trigger Consistency and Distribution*

Within a document, there is a strong trigger consistency: if one instance of a word triggers an event, other instances of the same word will trigger events of the same type[7].

There are also strong correlations among event types in a document. To see this we calculated the conditional probability (in the ACE corpus) of a certain event type appearing in a document when another event type appears in the same document.

| Event | Cond. Prob. |
|---|---|
| Attack | 0.714 |
| Transport | 0.507 |
| Injure | 0.306 |
| Meet | 0.164 |
| Arrest-Jail | 0.153 |

[7] This is true over 99.4% of the time in the ACE corpus.

| | |
|---|---|
| Sentence | 0.126 |
| Phone-Write | 0.111 |
| End-Position | 0.116 |
| Trial-Hearing | 0.105 |
| Convict | 0.100 |

**Table 3.1 - Events co-occurring with Die events with conditional probability > 10%**

As there are 33 subtypes, there are potentially 33·32/2=528 event pairs. However, only a few of these appear with substantial frequency. For example, there are only 10 other event types that occur in more than 10% of the documents in which a Die event appears. From **Table 3.1** and **Figure 3.1**, we can see that Attack, Transport and Injure events appear frequently with Die. We call these the related event types for Die.

The same thing happens for Start-Org events, although its distribution is quite different from Die events. For Start-Org, there are more related events like End-Org, Start-Position, and End-Position (**Figure 3.2**). But there are 12 other event types which never appear in documents containing Start-Org events.

From the above, we can see that the distributions of different event types are quite different, and these distributions might be good predictors for event extraction.

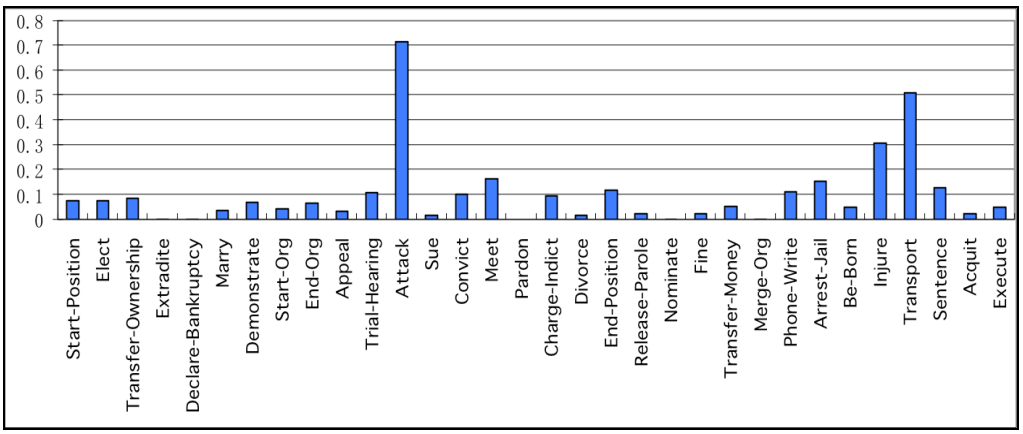


**Figure 3.1 - Conditional probability of the other 32 event types in documents where a Die event appears**

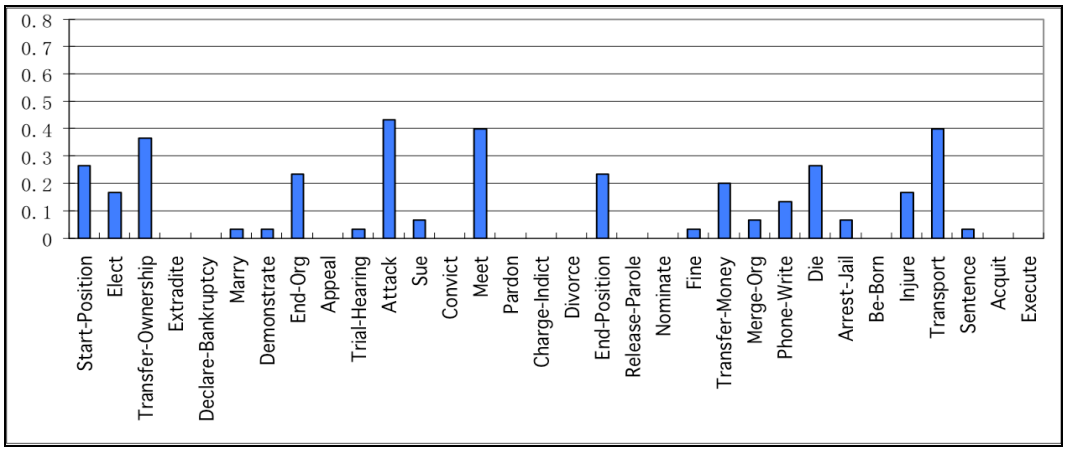


**Figure 3.2 - Conditional probability of the other 32 event types in documents where a Start-Org event appears**

### *3.1.2 Role Consistency and Distribution*

Normally one entity, if it appears as an argument of multiple events *of the same type* in a single document, is assigned the same role each time.[8]

Moreover, there is a strong relationship between the roles when an entity participates in different types of events in a single document. For example, we

[8] This is true over 97% of the time in the ACE corpus.

checked all the entities in the ACE corpus that appear as the *Target* role for an *Attack* event, and recorded the roles they were assigned for other event types. Only 31 other event-role combinations appeared in total (out of 237 possible with ACE annotation), and 3 clearly dominated. In **Figure 3.3**, we can see that the most likely roles for the *Target* role of the *Attack* event are the *Victim* role of the *Die* or *Injure* event and the *Artifact* role of the *Transport* event. The last of these corresponds to troop movements prior to or in response to attacks.

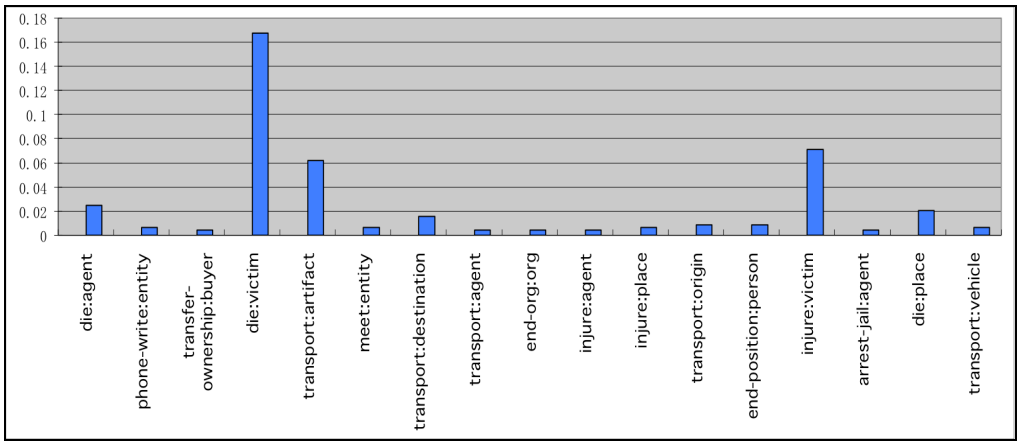


**Figure 3.3 - Conditional probability of all possible roles in other event types for entities that are the Targets of Attack events (roles with conditional probability below 0.002 are omitted)**

## *3.2 Document level Cross-event Approach*

In this section, we propose a document-level statistical model for event trigger and argument (role) classification to achieve document level within-event and cross-event consistency. Our event extraction system is a two-pass system where the sentence-level system is first applied to make decisions based on local information. Then the confident local information is collected and gives an

approximate view of the content of the document. The document level system is finally applied to deal with the cases which the local system can't handle, and achieve document consistency. To take advantage of cross-event relationships, we train two additional MaxEnt classifiers – a document-level trigger and argument classifier – and then use these classifiers to infer additional events and event arguments. In analyzing new text, the trigger classifier is first applied to tag an event, and then the argument (role) classifier is applied to tag possible arguments and roles of this event.

### *3.2.1 Confident Information Collector*

To use document-level information, we need to collect information based on the sentence-level baseline system. As it is a statistically-based model, it can provide a value that indicates how likely it is that this word is a trigger, or that the mention is an argument and has a particular role. We want to see if this value can be trusted as a confidence score. To this end, we set different thresholds from 0.1 to 1.0 in the baseline system output, and only evaluate triggers, arguments or roles whose confidence score is above the threshold. Results show that as the threshold is raised, the precision generally increases and the recall falls. This indicates that the value is consistent and a useful indicator of event/argument confidence (see **Figure 3.4**).

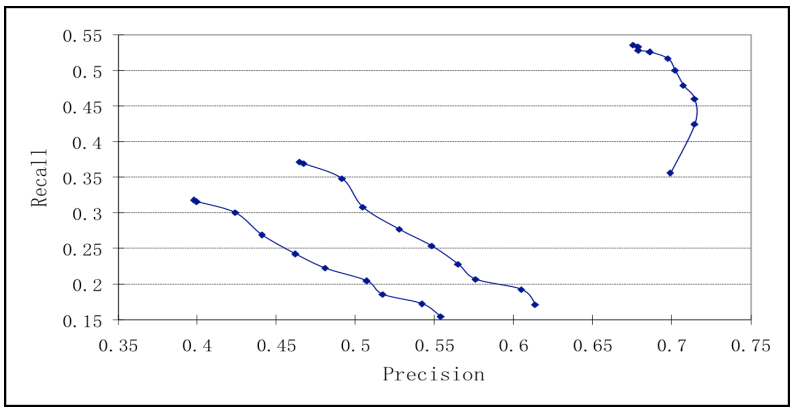


**Figure 3.4 -The performance of different confidence thresholds in the baseline system**
**on the development set**

To acquire confident document-level information, we only collect triggers and roles tagged with high confidence. Thus, a trigger threshold *t_threshold* and role threshold *r_threshold* are set to remove low confidence triggers and arguments. Finally, a table with *confident event* information is built. For every event, we collect its trigger and event type; for every argument, we use co-reference information and record every entity and its role(s) in events of a certain type.

To achieve document consistency, in cases where the baseline system assigns a word to triggers for more than one event type, if the margin between the probability of the highest and the second highest scores is above a threshold *m_threshold*, we only keep the event type with highest score and record this in the *confident-event* table. Otherwise (if the margin is smaller) the event type assignments will be recorded in a separate *conflict* table. The same strategy is applied to argument/role conflicts. We will not use information in the conflict table to infer the event type or argument/roles for other event mentions, because we

cannot confidently resolve the conflict. However, the event type and argument/role assignments in the conflict table will be included in the final output because the local confidence for the individual assignments is high.

As a result, we finally build two document-level confident-event tables: the event type table and the argument (role) table. A conflict table is also built but not used for further predictions (see **Table 3.2**).

| Event type table | |
|---|---|
| Trigger | Event Type |
| Met | Meet |
| Exploded | Attack |
| Went | Transport |
| Injured | Injure |
| Attacked | Attack |
| Died | Die |

| Argument role table | | |
|---|---|---|
| Entity ID | Event type | Role |
| 0004-T2 | Die | Time Within |
| 0004-6 | Die | Place |
| 0004-4 | Die | Victim |
| 0004-7 | Die | Agent |
| 0004-11 | Attack | Target |
| 0004-T3 | Attack | Time Within |
| 0004-12 | Attack | Place |
| 0004-10 | Attack | Attacker |

| Conflict table | | |
|---|---|---|
| Entity ID | Event type | Roles |
| 0004-8 | Attack | Victim, Agent |

**Table 3.2 - Example of document-level confident-event table (event type and argument role entries) and conflict table**

### *3.2.2 Document Level Classifiers*

We build two document level classifiers: one for trigger, and the other for argument.

**Document Level Trigger Classifier:** From the document-level confident-event table, we have a rough view of what kinds of events are reported in this document. The trigger classifier predicts whether a word is the trigger of an event, and if so of what type, given the information (from the confident-event table) about other types of events in the document. Each feature of this classifier is the conjunction of:

The base form of the word

An event type

A binary indicator of whether this event type is present elsewhere in the document (There are 33 event types and so 33 features for each word).

**Document level argument classifier:** The role classifier predicts whether a given mention is an argument of a given event and, if so, what role it takes on, again using information from the confident-event table about other events. As noted above, we assume that the role of an entity is unique for a specific event type, although an entity can take on different roles for different event types. Thus, if there is a conflict in the document level table, the collector will only keep the one with highest confidence, or discard them all. As a result, every entity is assigned a unique role with respect to a particular event type, or *null* if it is not an argument of a certain event type. Each feature is the conjunction of:

The event type we are trying to assign an argument/role to.

One of the 32 other event types

The role of this entity with respect to the other event type elsewhere in the document, or null if this entity is not an argument of that type of event

### *3.2.3 Document Level Event Tagging*

At this point, the low-confidence triggers and arguments (roles) have been removed and the document-level confident-event table has been built; the new classifiers are now used to *augment* the confident tags that were previously assigned based on local information.

For trigger tagging, we only apply the classifier to the words that do not have a confident local labeling; if the trigger is already in the document level confident-event table, we will not re-tag it.

The argument/role tagger is then applied to all events—those in the confident-event table and those newly tagged. For argument tagging, we only consider the entity mentions in the same sentence as the trigger word, because by the ACE event guidelines, the arguments of an event should appear within the same sentence as the trigger. For a given event, we re-tag the entity mentions that have not already been assigned as arguments of that event by the confident-event or conflict table.

## *3.3 Experiments*

We followed Ji and Grishman (2008)'s evaluation and randomly select 10 newswire texts from the ACE 2005 training corpora as our development set, which is used for parameter tuning, and then conduct a blind test on a separate set of 40

ACE 2005 newswire texts. We use the rest of the ACE training corpus (549 documents) as training data for both the sentence-level baseline event tagger and document-level event tagger.

| performance / system/human | **Trigger classification** | | | **Argument classification** | | | **Role classification** | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| Sentence-level baseline system | 67.6 | 53.5 | 59.7 | 46.5 | 37.2 | 41.3 | 41 | 32.8 | 36.4 |
| Within-event-type rules | 63 | 59.9 | 61.4 | 48.6 | 46.2 | 47.4 | 43.3 | 41.2 | 42.2 |
| Cross-event statistical model | 68.71 | 68.9 | **68.8** | 50.9 | 49.7 | **50.3** | 45.1 | 44.1 | **44.6** |
| Human annotation1 | 59.2 | 59.4 | 59.3 | 60.0 | 69.4 | 64.4 | 51.6 | 59.5 | 55.3 |
| Human annotation2 | 69.2 | 75.0 | 72.0 | 62.7 | 85.4 | 72.3 | 54.1 | 73.7 | 62.4 |

**Table 3.3 - Overall performance on blind test data**

To compare with previous work on within-event propagation, we reproduced Ji and Grishman (2008)'s approach for cross-sentence, within-event-type inference (see "within-event-type rules" in **Table 3.3**). We applied their within-document inference rules using the cross-sentence confident-event information. These rules basically serve to adjust trigger and argument classification to achieve document-wide consistency. This process treats each event type separately: information about events of a given type is used to infer information about other events of the same type.

We report the overall Precision (P), Recall (R), and F-Measure (F) on blind test data. In addition, we report the performance of two human annotators on 28 ACE newswire texts (a subset of the blind test set).[9]

From the results presented in **Table 3.3**, we can see that using the document level cross-event information, we can improve the F score for trigger classification by 9.0%, argument classification by 9.0%, and role classification by 8.1%. Recall improved sharply, demonstrating that cross-event information could recover information that is difficult for the sentence-level baseline to extract; precision also improved over the baseline, although not as markedly.

Compared to the within-event-type rules, the cross-event model yields much more improvement for trigger classification: rule-based propagation gains 1.7% improvement while the cross-event model achieves a further 7.3% improvement. For argument and role classification, the cross-event model also gains 3% and 2.3% above that obtained by the rule-based propagation.

The above experiments show that document-level information can improve the performance of a sentence-level baseline event extraction system. However, the model presented here is a simple two-stage recognition process; nonetheless, it has proven sufficient to yield substantial improvements in event recognition and event argument recognition. Richer models, such as those based on joint inference, may produce even greater gains

[9] The final key was produced by review and adjudication of the two annotations by a third annotator, which indicates that the event extraction task is quite difficult and human agreement is not very high.

# Chapter 4

# Document Level Topic Features[10]

As discussed in section 2.3, knowledge of the scenario is essential to identify the occurrence of an event, especially when local context is not sufficient. Given a narrow scope of information, even a human cannot make a confident decision. For example, for the sentence:

*(Ex 4–1) So he returned to combat …*

It is hard to tell whether it is an Attack event, which is defined as a violent physical act causing harm or damage, or whether it refers to a more innocent endeavor such as a tennis match. A broader field of view is often helpful to understand how facts tie together. If we read the whole article, and find it to be a terrorist story, it is easy to tag this as an Attack event; however, if it is in a tennis article, we probably won't tag it as an Attack event.

Most previous studies that acquire wider scope information use preselected corpora, like (Riloff 1996); or are rule-based, like Ji and Grishman (2008[11]); or involve supervised learning from the same training data, like Finkel et al. (2005), Liao and Grishman (2010a). However, such information depends mainly on the training data. We will be concerned in this chapter with situations where the test

[10] This chapter is mainly adapted from a published paper (Liao and Grishman, 2011b)

[11] See section 5.2 for more details.

data may not match the training, and so we are more interested in using an unsupervised topic model to provide such information.

There is not as much work on evaluation on a more balanced collection when the training corpus has a different distribution. Grishman (2010) first pointed out that understanding the characteristics of the corpus is an inherent parts of the event extraction task. He gave a small example of the effect of applying an event extractor to a more balanced corpus, and used a document classifier to reduce the spurious errors.

In this section, we propose to use a topic model (LDA) to provide document level topic information to solve this problem to some extent. In experiments, we not only evaluate its effect on ACE cross-validation, but also on a more balanced newswire corpus. Also, we compare this unsupervised approach to a supervised multi-label text classifier, and show that unsupervised topic modeling can get better results for both collections, and especially for a more balanced collection.

## *4.1 Data imbalance*

Why are we interested in unsupervised topic features? There is a problem that arises in the evaluation of almost all the tasks in NLP, concerning the similarity of distribution between training and testing data. Ideally, training and testing should be as similar as possible, but this is not always the case. In general, an effort is made to have the test corpora be representative of the sort of texts to which the NLP process is intended to be applied. In the case of the event extraction, this has generally been news sources such as newswires or broadcast news transcripts.

However, a particular event type is likely to occur infrequently in the general news, which might contain many different topics, only a few of which are likely to include mentions of this event type.

As a result, a typical evaluation corpus (a few hundred hand-annotated documents), if selected at random, would contain only a few events, which is not sufficient for training. To avoid this, these annotated corpora are artificially enriched through a combination of topic classification and manual review, so that they contain a high concentration of the events of interest. For example, in the MUC-3/4 test corpora, about 60% of the documents include relevant events, and in the ACE 2005 training corpus 48% include Attack events.

If we train and test the event extraction system on ACE annotated corpora, the problem is not significant because there are very few sports articles in the ACE evaluation: 74% of the instances of the word "combat" indicate an Attack event. However, if you extend the evaluation to a more balanced collection, for example, the un-filtered New York Times (NYT) newswire, you will find that there are a lot of sports articles and an event extractor will mistakenly tag lots of sports events as Attack events. Grishman (2010) drew attention to this phenomenon, pointing out that only about 17% of articles from the contemporaneous sample of The NYT newswire contained attack events, compared to 48% in the ACE evaluation. In this situation, if we apply the event extractor trained on the ACE corpus to the balanced NYT newswire, the performance may be significantly degraded.

## *4.2 Motivation*

Clearly, the topic of the document is a good predictor of particular event types. For example, a reference to "war" inside a business article might refer to a financial competition; while "war" inside a military article would be more likely to refer to a physical attack event. Text classification can be used here to identify document topic, and the final decision can be made based on both local evidence and document relevance. However, this method has three disadvantages:

First, the event type and document topic are not always strongly connected, and it depends significantly on what kind of event we are going to explore. If the events are related to the main category of the article, only knowing the article category is enough. But if they are not, treating each document as a single topic is not enough. For example, Die events might appear in military, financial, political or even sports articles. And most of the time, it is not the main event reported by the article. The article may focus more on the reason for the death, the biography of the person, or the effect of the death.

Second, when the article talks about more than one scenario, simple text classification will basically ignore the secondary scenario. For example, if a sports article that reported the results of a football game also mentions a fight between the fans of two teams, the topic of the document might be "sports", which is irrelevant to Attack events; however, there is an Attack event, which appears in the secondary scenario of the document.

Third, the category or relevance depends on the annotated data, and a classifier may be unable to deal with articles whose topics were rarely seen in the training data. Thus, if the category distribution of the evaluation data is different from the training data, a text classifier might have poor performance.

To solve the first two problems, we need to treat each document as a mixture of several topics instead of one; to solve the third problem, we want to see if unsupervised methods can give us some guidance which a supervised method cannot. These two goals are easily connected to a topic model, for example, Latent Dirichlet Allocation.

## *4.3 Topic Features*

A topic model, like Latent Dirichlet Allocation (LDA, David Blei, etc. 2003), is a generative model that allows sets of observations to be explained by unobserved groups. For example, if the observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word is attributable to one of the document's topics. For event extraction, there is a similar assumption that each document consists of various events, and each event is presented by one or several snippets in the document. We want to know if these two can be somehow connected and how one can improve the other.

We are more interested in an unsupervised approach from a large untagged corpus. In this way, we can avoid the data bias that may be introduced by an unrepresentative training collection, thus providing better high-level information

than previous approaches, especially when applied to the final target application instead of a specially selected development or evaluation corpus.

### *4.3.1 Unsupervised Features*

Latent Dirichlet Allocation (LDA) tries to group words into "topics", where each word is generated from a single topic, and different words in a document may be generated from different topics. Thus, each document is represented as a list of mixing proportions for these mixture components and thereby reduced to a probability distribution on a fixed set of topics. In LDA, each document may be viewed as a mixture of various topics. A document is generated by picking a distribution over topics, and given this distribution, picking the topic of each specific word to be generated. Then words are generated given their topics. Words are considered to be independent given the topics; this is a standard bag of words model assumption where individual words are exchangeable.

Unlike supervised classification, there are no explicit labels, like "finance" or "war", in unsupervised LDA. Instead, we can imagine each topic as "a cluster of words that refers to an implicit topic". For example, if a document contains words like "company", "financial", and "market", we assume it contains a "financial topic" and are more confident to find events like *Start-Position, End-Position*, while a document that contains "war", "combat", "fire", and "force" will be assumed to contain the "war topic", which is more likely to contain *Attack, Die*, or *Injure* events.

### *4.3.2 Supervised Features*

As the event extraction system uses a supervised model, it is natural to ask whether supervised topic features are better than unsupervised ones. There are several possible approaches. For example, we can first run a topic classification filter to predict whether or not a document is likely to contain a specific type of event. However, because of the limited precision of a simple classifier such as a bag-of-words MaxEnt classifier (for Attack events, the precision is around 69% in ACE data), using it as a pre-filter will lead to event recall or precision errors. Instead, we decide to use the topic information as features within the event extraction system. As one document might contain several event types, we tag each document with labels indicating the presence of one or more events of a given type, which is a multi-label text classification problem. In this section, we build a supervised multi-label text classifier to compare to the unsupervised topic model.

The basic idea for a multi-label classifier comes from the credit attribution problem in social bookmarking websites, where pages have multiple tags, but the tags do not always apply with equal specificity across the whole page (Ramage et al. 2009). This relation between tag and page is quite similar to that between event and document, because one document might also have multiple events of differing specificity. For example, an Attack event may be more related to the main topic of the document than a Meet event.

We use Labeled LDA (L-LDA) to build the multi-label text classifier, which is reported (Ramage et al. 2009) to outperform SVMs when extracting tag-specific document snippets, and is competitive with SVMs on a variety of datasets. L-LDA

associates each label with one topic in direct correspondence, and is a natural extension of both LDA and multinomial Naïve Bayes. In our experiment, each document can have several labels, each corresponding to one of the 33 ACE event types. In this way, we can easily map the goal of predicting the possible events in a document into a multi-label classification problem.

## *4.4 Experiment*

We set up two experiments to investigate the effect of topic information.

**ACE05 Experiment:** A 5-fold cross-validation on the whole ACE 2005 corpus, which contains 589 documents from June 2003.

**NYT03 Experiment:** An experiment to address the crucial issue of mismatch in topic distribution between training and test corpora. In this experiment, the ACE 2005 corpus is used as the training data, and unfiltered New York Times newswire data, a balanced corpus without pre-selection, is used for testing. The NYT corpus comes from the same epoch as the ACE corpus. This test data contains 75 consecutive articles[12].

### *4.4.1 Experiment Setup*

Encoding topic features into the baseline system is straightforward: as the occurrence of an event is decided in the final classifier – the trigger classifier – we add topic features to this final classifier. Although the argument / role classifiers

[12] We annotated the test data for the three most common event types in ACE – Attack, Die, and Meet – and evaluated this balanced corpus on these three events.

have already been applied, we can still improve the argument / role classification, because only when a word is tagged as a trigger will all the arguments/roles related to it be reported.

The unsupervised LDA was trained on the entire 2003 NYT newswire except for June to avoid overlap with the test data, a total of 27,827 articles; we choose K= 30, which means we treat the whole corpus as a combination of 30 latent topics.

The multi-label text classifier was trained on the same ACE training data as the event extraction, where each label corresponds to one event type, and there is an extra "none" tag when there are no events in the document. Thus, there are in total 34 labels.

For inference, we use the posterior Dirichlet parameters $\gamma^*(w)$ associated with the document (David Blei, etc. 2003) as our topic features, which is a fixed set of real-values. Thus, using the multi-label text classifier, there are 34 newly added features; while using unsupervised LDA, there are 30 newly added features. Stanford topic modeling software is used for both the multi-label text classifier and unsupervised LDA.

For preprocessing, we remove all words on a stop word list. Also, to reduce data sparseness, all inflected words are changed to their root form (e.g. "attackers"→"attacker").

### *4.4.2 Evaluation on ACE data*

We might expect supervised topic features to outperform unsupervised topic features when the distributions of the training and testing data are the same,

because its correlation to event type is clearer and explicit. However, it turns out not to be true in our experiment (see **Table 4.1**): the unsupervised features work better than the supervised features. This is understandable given that there are only hundreds of training documents for the supervised topic model, and the precision of the document classification is not very good, as we mentioned before in section 4.2. For unsupervised topics, we have a much larger corpus, and the topics extracted, although they may not correspond directly to each event type, predicate a scenario where a specific event might occur.

| Performance / System | **Trigger Classification** | | | **Argument Classification** | | | **Role Classification** | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| Baseline system | 64.3 | 51.1 | 56.9 | 69.4 | 21.8 | 33.2 | 62.8 | 19.7 | 30.0 |
| Multi-label classifier | 66.8 | 50.0 | 57.2 | 54.4 | 25.5 | 34.7 | 48.9 | 22.9 | 31.1 |
| Unsupervised LDA | 63.9 | 59.7 | **61.7** | 71.1 | 27.0 | **39.1** | 64.6 | 24.5 | **35.5** |

**Table 4.1 - Overall performance on ACE test data**

### *4.4.3 Evaluation on NYT data*

From the ACE evaluation, we can see that the unsupervised LDA works better than a supervised classifier, which indicates that even if the training and testing data are from the same distribution, the unsupervised topic features are more helpful. In our second evaluation, we evaluate on a more balanced newswire corpus, with no pre-selection.

First, we implement Grishman's solution (Simple Combination) to combine the document event classifier (a bag-of-words maximum-entropy model) with local evidence used in the baseline system. The basic idea is that if a document is classified as not related to a specific event, it should not contain any such events; while if it is related, there should be such events. Thus, an event will be reported if

$$\sqrt{P(reportable\_event) \times P(relevant\_document)} > \tau$$

where P(reportable_event) is the confidence score from the baseline system, while P(relevant_document) is computed from the document classifier.

**Table 4.2** shows that the simple combination method (geometric mean of probabilities) performs a little better than baseline. However, we find that the gains are unevenly spread across different events. For Attack events, it provides some benefit (from 57.9% to 59.6% F score for trigger labeling), whereas for Die and Meet events it does not improve much. This might be because Attack events are closely tied to a document's main topic, and using only the main topic can give a good prediction. But Die and Meet events are not closely tied to the document main topic, and so the simple combination does not help much.

Unsupervised LDA performs best of all, which indicates that the real distribution in the balanced corpus can provide useful guidance for event extraction, while supervised features might not provide enough information, especially when testing on a balanced corpus.

| Performance | **Trigger** | **Argument Classification** | **Role** |
|---|---|---|---|

| System | Classification | | | | | | Classification | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | R | F | P | R | F | P | R | F |
| Baseline system | 53.8 | 51.1 | 52.4 | 41.4 | 19.7 | 26.7 | 39.4 | 18.8 | 25.4 |
| Simple Combination | 63.1 | 47.4 | 54.2 | 41.4 | 19.7 | 26.7 | 39.4 | 18.8 | 25.4 |
| Multi-label classifier | 60.8 | 65.7 | 63.2 | 35.6 | 27.9 | 31.3 | 31.9 | 25.0 | 28.0 |
| Unsupervised LDA | 60.3 | 81.0 | **69.2** | 45.3 | 34.6 | **39.2** | 44.0 | 33.7 | **38.1** |

**Table 4.2 - Performance on NYT collection**

Our experiments indicated that an unsupervised document-level topic model trained on a large corpus yields substantial improvements in extraction performance and is considerably more effective than a supervised topic model trained on a smaller annotated corpus.

Finally, let us consider some examples to show why topic information helps so much on NYT data. First, we give an example where the supervised topics method does not work but unsupervised does. In our baseline system, many verbs in sports or other articles will be incorrectly tagged as Attack events. In such cases, as there are very few sports articles in ACE training data, and there is no event type related to sports, the supervised classifier might not capture this feature, and prefer to connect a sports article to an Attack event in the testing phase, because there are a lot of words like “shot”, “fight”. However, as there are a lot of sports articles in NYT data, the unsupervised LDA can capture this topic. Here is an example:

*(Ex 4–2) His only two shots of the game came in overtime and the goal was just his second of the playoffs, but it couldn't have been bigger.*

In (Ex 4–2), "shot" is tagged 67.5% of the time as an Attack event in training data. We checked the data and found that there are very few sports articles in the ACE corpus, and the word "shot" never appears in these documents. Thus, a supervised classifier will prefer to tag a document containing the word "shot" as containing an Attack event. However, because a sports topic can be explicitly extracted from an unannotated corpus that contains a reasonable portion of sports articles, the unsupervised model would be able to build a latent topic T which contains sports-related words like "racket", "tennis", "score" etc. Thus, most training documents which contain "shot" will have a low value of T; while the sports documents (although very few), will have a high value of T. Thus, the system will see both a positive feature value (the word is "shot"), and a negative feature value (T's value is high), and still has the chance to correctly tag this "shot" as not-an-event, while in the baseline system, the system will incorrectly tag it as an Attack event because there are only positive feature values.

The topic features can also help other event types. For Die events, consider:

*(Ex 4–3) A woman lay unconscious and dying at Suburban Hospital in Bethesda, Md.*

In (Ex 4–3), the word "dying" only appears 45.5% as a Die event in the training data, and is not tagged as a Die event by the baseline system. The reason is that there are a lot of metaphors that do not represent true Die events, like "dying nation", "dying business", "dying regime". However, when connected to the latent topic features, we know that for some topics, we can confidently tag it as a Die event.

For Meet events, we also find cases where topic features help:

*(Ex 4–4) President Bush meets Tuesday with Arab leaders in Egypt and the next day with the Israeli and Palestinian prime ministers in Jordan,….*

In (Ex 4–4), the baseline system misses this Meet event. The word "meets" only appears 25% of the time as a Meet event in the training data, because there are phrases like "meets the requirement", "meets the standard" which are not Meet events. However, adding topic features, we can correct this and similar event detection errors.

# Chapter 5

# Cross-Document IR-based Self-training[13]

In the preceding two chapters, we focused on improving a supervised event extraction system using information from wider scope. In this section, we investigate the effect of wider scope information on another task: semi-supervised learning (self-training) for event extraction. Annotating training data for event extraction is tedious and labor-intensive. Most current event extraction tasks rely on hundreds of annotated documents, but this is often not enough. Self-training is one way to automatically increase the training data, which can help us improve the event extraction system by adding more training samples.

[13] This chapter is mainly adapted from a published paper (Shasha Liao and Ralph Grishman, 2011a)

Self-training is a commonly used technique, where the classifier starts with a small amount of labeled data. The classifier is then used to classify the unlabeled data. Typically the most confident unlabeled points, together with their predicted labels, are added to the training set. The classifier is re-trained and the procedure repeated.

Self-training has been applied to several natural language processing tasks. For event extraction, there are several studies on bootstrapping from a seed pattern set. Riloff (1996) initiated the idea of using document relevance for extracting new patterns, and Yangarber et al. (2000, 2003) incorporated this into a bootstrapping approach, extended by Surdeanu et al. (2006) to co-training. Stevenson and Greenwood (2005) suggested an alternative method for ranking the candidate patterns by lexical similarities. Liao and Grishman (2010b) combined these two approaches to build a filtered ranking algorithm. However, these approaches were focused on finding instances of a scenario/event type rather than on argument role labeling. Starting from a set of documents classified for relevance, Patwardhan and Riloff (2007) created a self-trained relevant sentence classifier and automatically learned domain-relevant extraction patterns. Liu (2009) proposed the BEAR system, which tagged both the events and their roles. However, the new patterns were bootstrapped based on the frequencies of sub-pattern mutations or on rules from linguistic contexts, and not on statistical models.

Since the criterion for selecting the most confident examples is critical to the effectiveness of self-training, we need to first set up the metric to evaluate the confidence of an event. An event contains one trigger and an arbitrary number of

roles, and it is not easy to calculate its confidence as a whole. A confident trigger might contain some unconfident arguments, and a confident argument might refer to an unconfident trigger. Thus, we select a part of an entire event, containing one confident trigger and its most confident argument, to serve as one example and be fed back to the training system. In this way, for each mention $m_i$, its probability of filling a *role r* in a reportable event whose trigger is *t* is computed by:

$$P_{RoleOfTrigger}(m_i, r, t) = P_{Arg}(m_i) \cdot P_{Role}(m_i, r) \cdot P_{Event}(t)$$

where $P_{Arg}(m_i)$ is the probability from the argument classifier, $P_{Role}(m_i,r)$ is that from the role classifier, and $P_{Event}(t)$ is that from the trigger classifier. In each iteration, we added the most confident <role, trigger> pairs to the training data, and re-trained the system.

## *5.1 Motivation*

Surprisingly, traditional self-training does not perform very well (see **Table 5.1**) for events. The newly added samples do not improve the system performance; instead, its performance stays stable, and even gets worse after several iterations (see **Table 5.1**).

This reminds us of two problems with traditional self-training. First, as self-training uses its own predictions to teach itself, a classification mistake can reinforce itself. One way to avoid this is to stop bootstrapping if the prediction confidence drops below a threshold. However, this can only solve the problem to some extent. Another issue with self-training is that it always looks for the samples

most similar to the training data, and lacks the ability to finding "novel" examples, which can provide new information to learn.

We analyzed the data, and found that the poor performance is caused by these two common problems. First, we checked the accuracy of the confident samples predicted by our baseline system, and found that even the most confident samples are not always accurate. As a result, although self-training has been successful in other NLP tasks, like Named Entity Recognition, parsing, or part-of-speech tagging, where the baseline systems already have good performance, its performance on event extraction is not good, due to the baseline system's relatively poor performance. Thus, we need some techniques to filter the inaccurate samples out.

The problem of nothing "novel" being added is harder to solve. Co-training is a form of self-training which can address this problem to some extent. However, it requires two views of the data, where each example is described using two different feature sets that provide different, complementary information. Ideally, the two views are conditionally independent and each view is sufficient (Zhu, 2008). Co-training has had some success in training (binary) semantic relation extractors for some relations, where the two views correspond to the arguments of the relation and the context of these arguments (Agichtein and Gravano 2000). However, it has had less success for event extraction because event arguments may participate in multiple events in a corpus and individual event instances may omit some arguments.

## *5.2 Cross-Document IR-based Approach*

Since word sense disambiguation is already a hard task, and we do not have any annotated data to do word sense disambiguation on the event level, we want to find another way out. Yarowsky (1995) introduced the idea of sense consistency, declaring that the same words in the same document are likely to share the same senses. Later, he extended this idea to operate across related documents. Yangarber et al. (Yangarber and Jokipii, 2005; Yangarber, 2006; Yangarber et al., 2007) applied cross-document inference to correct local extraction results for disease name, location and start/end time. Mann (2007) encoded specific inference rules to improve extraction of information about CEOs (name, start year, end year).

There also are some studies on using this idea to improve the performance of event extraction. Ji and Grishman (2008) incorporated this idea to propagate consistent triggers and arguments across topic-related documents using rule-based inference. Gupta and Ji (2009) used a similar approach to recover implicit time information for events. Liao and Grishman (2010a) use a statistical model to infer the cross-event information within a document to improve event extraction. However, these earlier studies use rule-based inference or supervised approach, and none apply this hypothesis to aid a semi-supervised approach. In the following sections, we will explain how the one-sense-per-discourse hypothesis can be applied to improve the self-training and why it helps.

### *5.2.1 Self-training on Information Retrieval Selected Corpus (ST_IR)*

To address the first problem (low precision of extracted events), we tried to select a corpus where the baseline system can tag the instances with greater confidence. Ji and Grishman (2008) have observed that the events in a cluster of documents on the same topics as documents in the training corpus can be tagged more confidently. Thus, we believe that bootstrapping on a corpus of topic-related documents should perform better than a regular newswire corpus.

We followed Ji and Grishman (2008)'s approach and used the INDRI retrieval system[14] (Strohman et al., 2005) to obtain the top N related documents for each annotated document in the training corpus. The query is event-based to insure that related documents contain the same events. For each training document, we construct an INDRI query from the triggers and arguments. For the following sentence:

*(Ex 5–1) Bob Cole* ***was*** *killed* ***in*** *France today****;*** *he* ***was*** *attacked****…***

query keywords will include "killed", "attacked", "France", "Bob Cole", and "today". Only names and nominal arguments will be used; pronouns appearing as arguments are not included. For each argument we also add other names coreferential with the argument.

[14] http://www.lemurproject.org/indri/

### *5.2.2 Self-training using Global Inference (ST_GI)*

Although bootstrapping on related documents can solve the problem of "confidence" to some extent, the "novelty" problem still remains: the top-ranked extracted events will be too similar to those in the training corpus. To address this problem, we propose to use a simple form of global inference based on the special characteristics of related-topic documents. Previous studies pointed out that information from wider scope, at the document or cross-document level, could provide non-local information to aid event extraction (Ji and Grishman 2008, Liao and Grishman 2010a). There are two common assumptions within a cluster of related documents:

***Trigger Consistency Per Cluster:*** if one instance of a word triggers an event, other instances of the same word will trigger events of the same type.

***Role Consistency Per Cluster:*** if one entity appears as an argument of multiple events of the same type in a cluster of related documents, it should be assigned the same role each time.

Based on these assumptions, if a trigger/role has a low probability based on the local context, but a high probability based on another place in the article, it means that the local context of this trigger/role tag is not frequently seen in the training data, but the tag is still confident. Thus, we can confidently add it to the training data and it can provide novel information which the samples confidently tagged by the baseline system cannot provide.

To start, the baseline system extracts a set of events and estimates the probability that a particular instance of a word triggers an event of that type, and

the probability that it takes a particular argument. The global inference process then begins by collecting all the confident triggers and arguments from a cluster of related documents[15]. For each trigger word and event type, it records the highest probability (over all instances of that word in the cluster) that the word triggers an event of that type.

First, for each argument, within-document and cross-document coreference are both applied to collect all instances of that entity. We use a statistical within-document coreference system (Grishman et al. 2005) to acquire coreference inside one document, and a simple rule-based cross-document coreference system, where entities sharing the same names will be treated as coreferential across documents.

Then, we compute the maximum probability (over all instances) of that argument playing a particular role in a particular event type. These maxima will then be used in place of the locally-computed probabilities in computing the probability of each trigger-argument pair in the formula for PRoleOfTrigger given above[16]. For example, if the entity "Iraq" is tagged confidently (probability > 0.9) as the "Attacker" role somewhere in a cluster, and there is another instance where from local information it is only tagged with 0.1 probability to be an "Attacker" role, we use probability of 0.9 for both instances.

[15] In our experiment, only triggers and roles with probability higher than 0.9 will be extracted.

[16] If a word or argument has multiple tags (different event types or roles) in a cluster, and the difference in the probabilities of the two tags is less than some threshold, we treat this as a "conflict" and do not use the conflicting information for global inference.

In this way, a trigger pair containing this argument is more likely to be added into the training data through bootstrapping, because we have global evidence that this role probability is high, although its local confidence is low. In this way, some novel trigger-argument pairs will be chosen, thus improving the baseline system.

Here is an example:

*(Ex 5-2)* ***Miroslay Kostelka*** *was* ***named*** *as new Czech defense minister Monday…*

*(Ex 5-3) It was unclear how the* ***Kostelka appointment*** *would affect the reform plans.*

*(Ex 5-4) Prime Minister Vladimir Spidla said Monday after* ***Kostelka****'s* ***appointment*** *that…*

Ex 5-2 is extracted from the annotated document, and thus the probability that "Miroslay Kostelka" is the Person role of Start-Position event (triggered by "named") is 1.0. However, the probability that "Kostelka" is the Person role of a Start-Position event in Ex 5-3 and Ex 5-4 is 0.745 and 0.794 respectively. Thus, Ex 5-3 and 5-4 will not be added to the training data in bootstrapping; however, using global information to adjust local probability, these confident and novel samples will be selected in bootstrapping.

## *5.3 Experiments*

We randomly chose 20 newswire texts from the ACE 2005 training corpora (from March to May of 2003) as our evaluation set, and used the remaining newswire texts as the original training data (83 documents). For self-training, we

picked 10,000 consecutive newswire texts from the TDT5 corpus from 2003[17] for the ST experiment. For ST_IR (see section 5.2.1) and ST_GI (see section 5.2.2), we retrieved the best N (using N = 25, which (Ji and Grishman 2008) found to work best) related texts for each training document from the English TDT5 corpus consisting of 278,108 news texts (from April to September of 2003). In total we retrieved 1650 texts; the IR system returned no texts or fewer than 25 texts for some training documents. In each iteration, we extract 500 trigger and argument pairs to add to the training data.

**Table 5.1** shows that bootstrapping on an event-based IR corpus can produce improvements on all three evaluations, while global inference can yield further gains.

| | **Trigger labeling** | **Argument labeling** | **Role labeling** |
|---|---|---|---|
| Baseline | 54.1 | 39.2 | 35.4 |
| ST | 54.2 | 40.0 | 34.6 |
| ST_IR | 55.8 | 42.1 | 37.7 |
| ST_GI | **56.9** | **43.8** | **39.0** |

**Table 5.1 - F-score with different self-training strategies after 10 iterations**

[17] We selected all bootstrapping data from 2003 newswire, with the same genre and time period as ACE 2005 data to avoid possible influences of variations in the genre or time period on the bootstrapping. Also, we selected 10,000 documents because this size of corpus yielded a set of confidently-extracted events (probability > 0.9) roughly comparable in size to those extracted from the IR-selected corpus; a larger corpus would have slowed the bootstrapping.

Experiments show that using an IR-selected corpus improves trigger labeling F score 1.7%, and role labeling 2.3%. Global inference can achieve further improvement of 1.1% for trigger labeling, and 1.3% for role labeling. Also, this bootstrapping involves processing a much smaller but more closely related corpus, which is more efficient. Such pre-selection of documents may benefit bootstrapping for other NLP tasks as well, such as name and relation extraction.

# Chapter 6

# Sentence Level Active Learning[18]

In the previous chapter, we investigated the wider scope features from a document or cross-document level; in this chapter, we explore the usage of features from a narrower scope: sentence level. Under this approach, we treat the entire sentence as a whole, and introduce its effect in the active learning approach.

Active learning is a supervised machine learning technique in which the learner is in control of the selection of data used for learning. The intent is to ask an oracle - typically a human with extensive knowledge of the domain at hand - about the classes of instances for which the model trained so far makes unreliable predictions. Selective sampling methods, introduced by Cohn, Atlas and Ladner (1994), made the learner query the oracle about data that is likely to be

[18] This chapter is mainly adapted from a published paper (Shasha Liao and Ralph Grishman, 2011c)

misclassified. This is the crucial aspect of AL - finding "good" instances for a human to annotate.

Many existing active learning methods are based on selecting the most uncertain examples using various measures (Thompson et al. 1999; Schohn and Cohn 2000; Tong and Koller 2001a; Tong and Koller 2001b; Engelson and Dagan 1999). (McCallum and Nigam 1998; Tang et al. 2002) proposed methods that consider the representativeness criterion in active learning. Tang et al. (2002) used the density information to weight the selected examples but do not use it to select a sample. Brinker (2003) first incorporated diversity in active learning for text classification. Shen et al. (2004) proposed a multi-criteria-based active learning approach and applied it to named entity recognition. They jointly consider multiple criteria, including *informativenes*s, *representativeness* and *diversity*. Experiments showed that incorporating all the criteria together is more efficient than single-criterion-based methods.

Traditional active learning with redundant views splits the feature set into several sub-sets or views, each of which is enough, to some extent, to describe the underlying problem. Muslea et al. (2000) presented an approach in which two classifiers are trained only on labeled data, then run over the unlabeled data. A contention set of examples is then created, consisting of all unlabeled examples on which the classifiers disagree. Samples are randomly selected from this set for query, and then both classifiers are retrained.

To the best of our knowledge, there is no study yet of active learning in event extraction. However, Patwardhan and Riloff (2009) presented a model for role

filling in event extraction that jointly considers both the local context around a phrase and the wider sentential context in a probabilistic framework. They used a sentential event recognizer and a plausible role-filler recognizer to jointly make decisions on a sentence, and find the roles of the events. Although it is not a co-testing process, it gave us the intuition of using a sentential view to predict possible events in a sentence. Surdeanu et al. (2006) used a co-training strategy in which two classifiers seek to classify documents as relevant to a particular scenario. In his approach, a bag of words model was used to determine relevance for event extraction. However, his work differs from ours because it involves semi-supervised learning, and it uses a document-level classifier instead of a sentence-level classifier. Patwardhan and Riloff (2007) presented an information extraction system that find relevant regions of text and applies extraction patterns within those regions. They created a self-trained relevant sentence classifier to identify relevant regions, and use a semantic affinity measure to automatically learn domain-relevant extraction patterns. They also distinguish primary patterns from secondary patterns and apply the patterns selectively in the relevant regions.

After studying several sampling strategies, we settled upon a *pseudo co-testing* approach where a second classifier that solves a coarser variant of the original task is used. Furthermore, we incorporate multiple selection criteria into the pseudo co-testing, not only selecting more informative sentences, but also considering their distribution in the sample pool, and the diversity of the instances added to the training set at the same time. Experiments show that a classifier for a coarser task can provide an extra view to build a pseudo co-testing strategy. Although the

ultimate goal involves training the original (fine-grained) classifier, the coarser task can provide useful information for query selection. In the special case of event extraction, we find that a sentence classifier can help an event tagger select a better query, because it is not only good at finding new trigger and local structures from graded matching over a wider scope, but also provides a better way of judge the representativeness and diversity of the samples. In our experiment, we reduced human labor by 80.6% to 87.8%.

In particular, we apply the active learning approach to the *Attack* event, because it is the most frequent event type in the ACE corpus, and is particularly challenging because of the large number of different expressions: there are 312 different words in the corpus that serve at least once as the trigger of an *Attack* event.

## *6.1 Motivation*

Annotating a corpus in order to train an event tagger is a costly task. First of all, event extraction is difficult and requires substantial training data. The same event might be presented in various expressions, and an expression might represent different events in different contexts. For example, “retire” and “resign” can both represent an End-Position event, while “leave” can represent either an End-Position or Move event in different contexts. Moreover, for each event type, the event participants and attributes may also appear in multiple forms and exemplars of the different forms may be required.

Furthermore, compared to other tasks like name tagging or part of speech tagging, events of a particular type appear relatively rarely in a document. One document might only contain one or two events of a given type, or even none at all. For the ACE 2005 event extraction task, Attack events have the highest frequency in the training corpus (2240 times, an average of 4 events per document), while Start-Position events only appear 232 times (an average of 1/3 event per document). As a result, to acquire enough training samples, we need to annotate a lot of documents. If we can predict which documents, or even which sentences to annotate, we can save a lot of time.

Considering the complexity of event extraction and the labor of annotating an event, providing the annotator with an informative sample to annotate is especially important. Active learning (AL) is a good way to do so because it aims to keep the human annotation effort to a minimum, only asking for advice where the training utility of the result of such a query is high.

## *6.2 Pseudo Co-testing Approach*

Active learning has been successfully applied to a number of natural language processing tasks, such as named entity recognition (Shen et al. 2004; Hachey, Alex and Becker 2005; Kim et al. 2006), text categorization (Schohn and Cohn 2000; Tong and Koller 2002; Hoi, Jin & Lyu 2006), part of speech tagging (Ringger et al. 2007), parsing (Osborne and Baldridge 2004; Becker and Osborne 2005; Reichart and Rappoport 2007), and word sense disambiguation (Chen et al. 2006;

Zhu and Hovy 2007). However, there have not yet been any studies to use active learning in event extraction.

There are several sampling methods in active learning; the most commonly used ones include uncertainty-based sampling, committee-based sampling, and co-testing. Co-testing Muslea et al. (2000) involves two (or more) redundant views; it simultaneously trains a separate classifier for each view, and the system selects a query based on the degree of disagreement among the learners. Because well-informed classifiers for the two views should agree, co-testing will select an example that is informative for at least one of the classifier models.

The advantage of uncertainty sampling is that it is simple and can be applied to almost all kinds of statistical models. However, Muslea (2000) points out that uncertainty sampling may make queries that lead to minimal improvements of the classifier, and therefore require more queries to build an accurate classifier.

The advantage of co-testing is that it has better performance than uncertainty based sampling. The disadvantage is that it has more constraints: the two views should be disjoint and each sufficient to learn a classifier. As discussed above, event extraction is complicated and involves several classifiers on different levels interacting together. This makes it difficult to split the feature set into two views. In particular, the identity of the trigger will be a critical feature for any successful classifier. Committee-based sampling faces similar problem as co-testing: it is hard to generate several classifiers that are consistent with the training set or sub-samples of it, respectively.

We could do active learning at the token level – asking the *oracle* whether a specific token triggers an event – but that is not very practical. Rather, for each query, we return a sentence that might contain an event to ask the oracle to annotate. We do so because the oracle needs to read the whole sentence to decide whether it is a reportable event, and annotate all its arguments. Thus, a sentence-based sampling pool is built where each sentence is treated as a sample query.

### *6.2.1 Applying Uncertainty-based Sampling*

Event extraction is a compound classification task, which involves the identification of arguments/roles, and the event trigger. These classifiers are separately trained, but not independent; results from previous classifiers are used as features for the following classifier, and the decision by the following classifier will affect the previous results (arguments confidently tagged by the argument/role classifier will be discarded if the trigger labeling treats it as not a event). Because the final classifier – the trigger classifier – takes all the considerations we mentioned above as input, and makes a final decision of a reportable event, we use its output as the probability of the event tagger. The traditional approach in uncertainty sampling (Lewis and Gale 1994) queries one of the samples on which the classifier is the least confident. In our case, the greatest uncertainty regarding the presence of an event corresponds to the trigger probability closest to 0.5. We treat the uncertainty of the sentence as the maximum of the uncertainties of the constituent words (i.e., the uncertainty attributable to the word with probability closest to 0.5):

$$e\_Info(S_i) = \quad 1 - \min_{w_j \in S_i} \left| 0.5 - prob\_e(w_j) \right|$$

where *prob_e(w$_j$)* is the trigger probability of the word $w_j$ in $S_i$ , as returned by the event tagger.

### *6.2.2 Problems with Uncertainty-based Sampling*

However, the results of uncertainty-based sampling are somewhat disappointing (see Figure 6.2, Figure 6.3 and Figure 6.4). It performs quite well at first: within a few iterations, trigger labeling (event detection) quickly achieves a performance (F score) of 65%, but beyond that point the gain is very slow. At this point there is still a 7% gap between its performance and that of a classifier trained on the whole sampling pool.

Why does uncertainty-based AL perform this way? The event tagger depends primarily on the particular trigger and secondarily on its local structure, for example, the potential arguments in the immediate vicinity of the trigger and the dependency paths between them. Such information is effective at identifying the trigger and arguments, but is responsive only to particular words and patterns. Triggers and structures that have not been seen in the training data will be assigned uniformly low probabilities. When trained on the whole ACE 2005 corpus (in a supervised training scenario) this is appropriate behavior: we don't want to report an event in testing if we haven't seen the trigger before.[19] However, for active learning, the inability to differentiate among potential new triggers and local

[19] Unlike some other tasks such as named entity and part-of-speech tagging, local contextual clues by themselves are generally not strong enough to reliably tag an event.

structures is critical. Only a few words ever serve as possible triggers for a specific event type. For the *Attack* event, only 2.0% of the words in the ACE training data ever act as an event trigger. The uncertainty of the event tagger, by itself, does not provide useful guidance regarding possible additional triggers the user should be asked about, and the system might query a lot of irrelevant sentences with unseen words before a sentence with a new trigger is found.

We can see this as an instance of a more general problem. Our goal in AL is to select for labeling those data points that are most likely to improve the accuracy of the model. Methods like uncertainty-based sampling are heuristics towards that end, but are not always effective; their success depends on characteristics of the classifier and the feature space. For event extraction, the classifier is most likely to benefit from finding *new*, *frequently-occurring* triggers. We need a way of identifying likely candidates.

Furthermore, we note that – while the final trigger classifier that we train from the labeled data must operate at the token level – we will be presenting the user with a sentence to label, so it is sufficient for the classifier we use for AL to operate at the sentence level.

### *6.2.3 Another View from Sentential Scope*

Can we find a classifier that suits the needs of our active learner by identifying sentences which are likely to contain an event? A simple (bag-of-words) classifier based on the words in the sentence can do quite well at this task. For example, a sentence with "troops", "victim", "bloody" and "soldier" might be more likely to contain an *Attack* event, even if these words might not be elements of the event.

These bag-of-words features are not particularly helpful for the original task of identifying an event (trigger and arguments) – they don't pinpoint a particular word as the trigger. But that's not a problem if the data selection for AL is operating at a coarser level.[20]

### *6.2.4 Pseudo Co-Testing*

The sentence-level bag-of-words classifier is far from perfect – the predictions at the sentence level are somewhat noisy. But considering that only 6.5% of the sentences in the ACE data contain an *Attack* event, returning a possibly relevant sentence is much more useful than returning a totally irrelevant sentence. If a sentence *S* in the sample pool shares many words with another sentence in the training data known to contain an event, and the event tagger does not find a trigger word in *S*, there is a good chance that *S* contains a new (previously unseen) trigger word and new local structure, because the two sentences may be describing the same event, but using different verbs and word sequences.

Thus, we apply a pseudo co-testing algorithm with one view from an event tagger based on local information, and another view, which aimed to solve an approximate task: whether there is a possible event in a sentence.

---

[20] Note that some active learners for tasks such as named entities and part-of-speech which also train token-level annotators choose to present data to the user at the sentence level, because it is more convenient and efficient for the user. These taggers could select data at the token level using two views based on the identity of a token and its immediate context. We share these user considerations, but in addition selecting data at the sentence level enables us to create effective complementary views for event extraction not available at a finer (token) level.

We call this algorithm "*pseudo* co-testing" because one of the views is not sufficient to solve the target problem, but is sufficient to solve a subproblem at a coarser granularity, in contrast to traditional co-testing. People might argue that when a *pseudo contention point* is found in this algorithm, it means that at least one of the classifiers is wrong, but we do not know (until we query the oracle) which one. If it is the event tagger, this sample is informative for the event tagger and adding this sample will improve the performance; if it is the sentence classifier, it is not guaranteed that this sample is informative for the event tagger. However, since the updated sentence classifier will serve to select subsequent queries, samples informative for the sentence classifier should accelerate subsequent active learning. Furthermore, the event tagger and the sentence classifier each have their own advantages in finding an event to query. The event tagger prefers sentences with already-known local patterns, like a trigger and its arguments, although the overall sentence (the choice of words and wider structure) might be very different. The sentence classifier prefers sentences sharing the same words, but which may have different local structures. Together they offer the potential for finding new triggers that do not appear in the existing training data (via the sentence classifier) and then acquiring event and non-event exemplars of these triggers (through the event tagger).

In pseudo co-testing, we use the probabilities from the event tagger and sentence classifier to build a contention set consisting of those sentences where the event tagger and sentential event recognizer make different predictions. Among these sentences, we assume that the larger the margin between the event tagger and

sentential event recognizer, the less certain the sample is. So, instead of randomly choosing samples from the contention set, we order the samples by their margins between the event tagger and sentential event recognizer, and pick the ones with largest margin:

$$co\_Info(S_i) = \underset{w_j \in S_i \& isCP}{Max} \left| prob\_e(w_j) - prob\_s(S_i) \right|$$

where $prob\_s(S_i)$ is the probability from the sentence classifier; while $prob\_e(w_j)$ is the trigger probability from the event tagger for the word $w_j$ in sentence $S_i$, and $w_j$ is a *contention point* (CP) where the event tagger's prediction is opposite that of the sentence classifier.

## *6.3 Multi-criteria-based AL*

Normally active learning only considers the *informativeness* of the sample. In uncertainty-based query, *informativeness* is represented by the least confident sample; in committee-based querying, it is represented by the samples on which the committee vote is the most equally split; in co-testing, it is represented by the contention sample. Shen et al. (2004) pointed out that we should maximize the contribution of the selected instances based on multiple criteria besides *informativeness*. For example, the *representativeness* and *diversity* of the sentence should also be considered. In this way, we not only consider whether the current model contains enough information to classify this sentence (as containing an event), but also consider the distribution of this sample in the whole sampling pool

(*representativeness*), and moreover, insure that we select different kind of samples in a batch to make the selection more diverse (*diversity*).

### *6.3.1 Features used in Similarity of Samples*

To evaluate the *representativeness* and *diversity*, we first need to calculate the similarity between two samples, in our case, two sentences. In general, a sentence will be represented as a vector of features $S_1 = \{f_{11}, f_{12}, f_{13}, ....., f_{1n}\}$ and the similarity is calculated based on the feature vectors of the two sentences. Thus, the essential problem becomes how to build the feature vector for a sentence. Since there are two classifiers in the pseudo co-testing, we use features from both classifiers, and measure the similarity using a cosine measure, following Shen et al (2004):

$$Sim(S_1, S_2) = \frac{\sum_{fi \in S_1} \sum_{f_j \in S_2} sim(f_i, f_j)}{|S_1| \quad |S_2|}$$

where $sim(f_i, f_j)$ is 1 when $f_i$ and $f_j$ are the same, otherwise 0.

### *6.3.2 Representativeness*

A few prior studies have considered this selection criterion (McCallum and Nigam 1998; Tang et al. 2002; Shen et al. 2004). The *representativeness* of a sample can be evaluated based on how many samples are similar to this sample. Adding samples which are more representative to the training set will have an effect on a larger number of unlabeled samples.

For every sentence in the sampling pool, we measure its *representativeness* based on its average similarity to other sentences in the sampling pool:

$$Represent(S_i) = \frac{\sum_{S_j \in P, i \neq j} sim(S_i, S_j)}{|P| - 1}$$

where $P$ is the current sampling pool. In this way, we will filter out the samples that are rare in the whole sampling pool, and focus our effort on the samples that appear more frequently in the whole corpus.

In addition to favoring the most informative example, we also prefer the most representative example. To combine scores from *informativeness* and *representativeness*, we followed Shen et al (2004)'s metric:

$$Score(S_i) = \lambda \cdot co\_Info(S_i) + (1 - \lambda) Represent(S_i)$$

where the relative importance of each criterion is determined by the parameter $\lambda$ ($0 \leq \lambda \leq 1$). In our experiment, $\lambda$ is set to 0.7.

### *6.3.3 Diversity*

The role of the diversity criterion is to maximize the training utility of a batch of samples. As we add a batch of samples into the training data in one iteration (for efficiency in updating the model), we want to make sure we provide various types of sentences, which provide the most information as a whole, and avoid selecting very similar sentences for a single batch. To this end, after we rank the sentences in the sampling pool, based on the different strategies mentioned above, we skip over any sentence whose similarity to one already selected in the same batch exceeds a threshold (see Figure 6.1).

The diversity metric is involved in selecting a batch of instances, as follows:

```
==================
Given: SenSet = (S1,...,SN) and the BatchSet with the
maximal size K.
Initialization: BatchSet = empty
Loop until BatchSet is full
        Select Si based on some measure from SenSet;
        RepeatFlag = false;
        Loop from j = 1 to CurrentSize of BatchSet
                If Score(Si, Sj) > threshold Then
                        RepeatFlag = true;
                        break;
        If RepeatFlag == false Then
                Add Si into BatchSet.
====================
```

**Figure 6.1 - Diversity criterion in batch-based active learning**

## *6.4 Experiments*

In the following sections, we compare the performance of the query strategies mentioned above – uncertainty-based query (*Uncertainty*), pseudo co-testing (*pCT*), and multi-criteria pseudo co-testing (*multi_pCT*). We employ a random sampling (*Random*) method as a baseline, where samples are selected randomly to add to the training data. Also, to assess the benefit of active learning, we report the performance from the event tagger trained on the entire ACE2005 data except for the test set (*Full_Corpus*).

We use the ACE 2005 training corpus, which contains in total 598 annotated documents, to simulate the active learning process. For evaluation, we conduct a blind test on a set of 54 randomly chosen documents. For each active learning strategy, we make 4 runs and use the average scores as our final results. For each run, 10 documents are randomly chosen as the initial training data, and the rest (534 documents) are used to build the sampling pool. Overall, the average initial

training set contains 369 sentences, and the sampling pool contains an average of 12074 sentences.

A Maxent model based on bag-of-words features serves as the sentence classifier. To reduce data sparseness, all inflected words are changed to their lemma form (e.g. “attackers”→“attacker”). A list of stop words is also applied.

In each iteration, we picked 50 sample sentences at the top of the ranked list based on different query strategies. To simulate the user queries, annotations extracted from the key annotations are returned as user feedback, and added into the training data.

The performances (F-measure) of different strategies are evaluated based on three metrics: argument/role labeling (**Figure 6.2**, **Figure 6.3**) and trigger labeling (**Figure 6.4**).

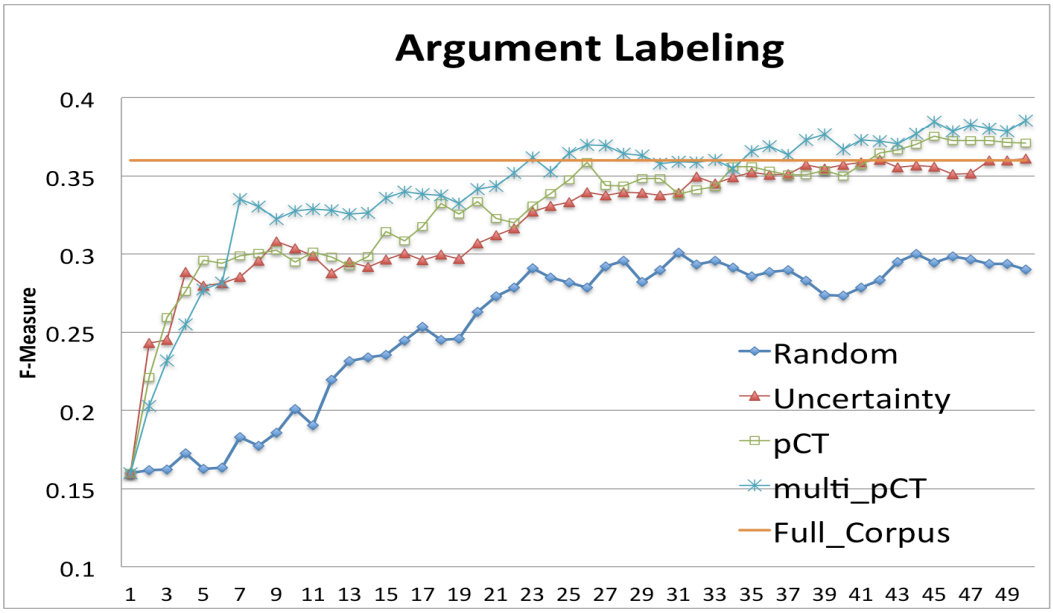


**Figure 6.2 - Performance (F-Measure) of argument labeling**

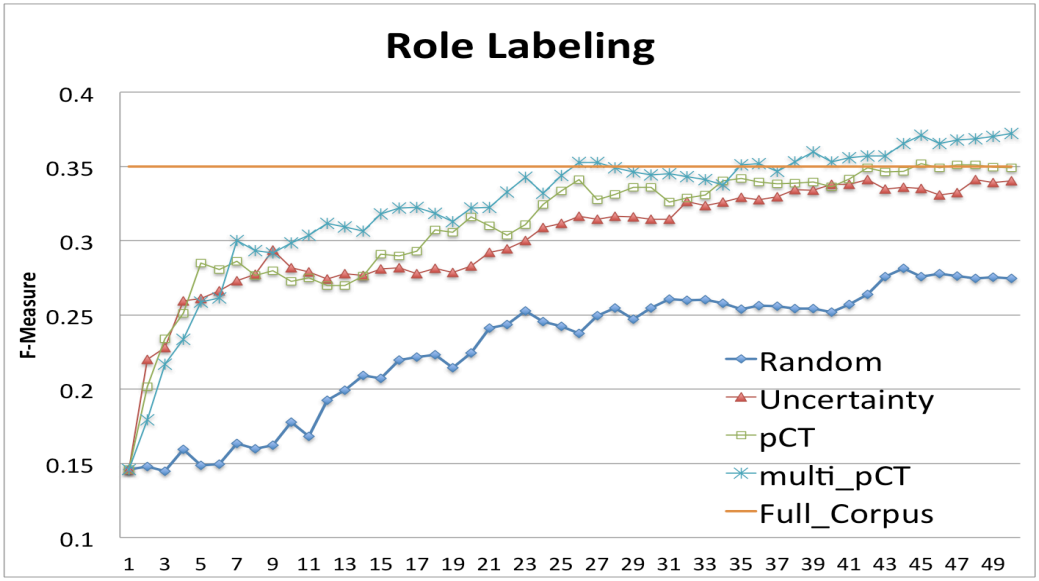


**Figure 6.3 - Performance (F-Measure) of role labeling**

Uncertainty-based querying (*Uncertainty*) yields poorer results than the other active learning strategies, because of the event tagger's relatively rigid matching procedure. Thus, it lacks the ability to recognize new potential triggers or patterns. For example, if we have pattern *A* which is very similar to some event-bearing patterns in the training data, and pattern *B* which is quite different from any pattern in the training data, the event tagger will treat them the same. However, the sentence classifier provides more graded matching, and gives the sentence containing pattern *A* higher score because they share a lot of words. Thus, the pseudo co-testing (*pCT*) would give a higher score to pattern *A*, and achieve better performance. Also, we observed that multi-criteria pseudo co-testing (*multi_pCT)* performs best in all three evaluations.

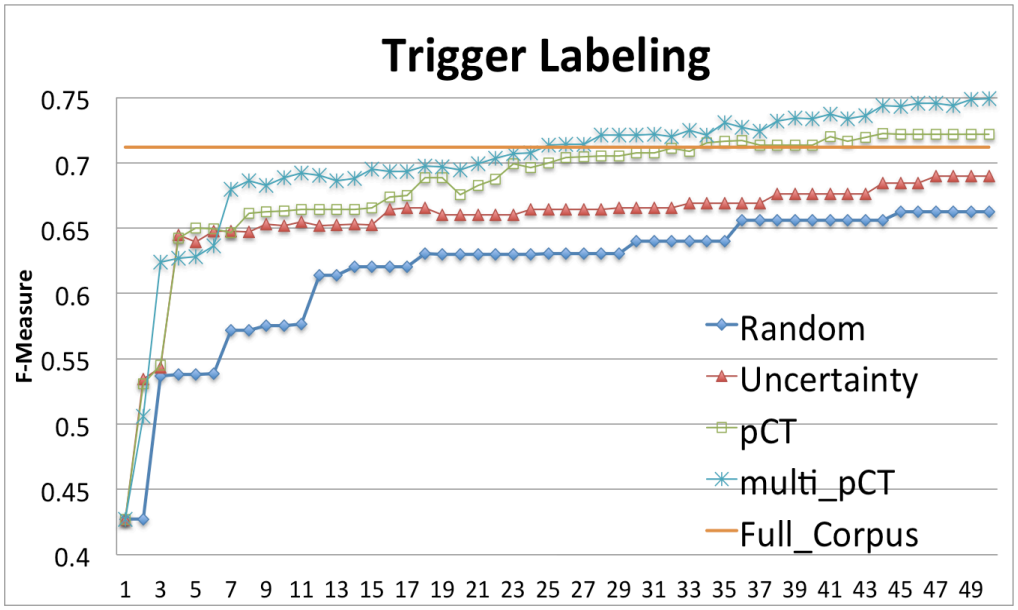


**Figure 6.4 - Performance (F-Measure) of trigger labeling**

The differences between the approaches are particularly marked for trigger labeling after just a few iterations. Consider how much data must be annotated to get to 95% of full corpus score for trigger labeling (F-Measure 67.5%): *multi_pCT* only takes 7 iterations; *pCT* takes 17 iterations; *Uncertainty* takes 38 iterations. In other words, 5.8%, 9.8%, 18.2% of the whole corpus needs to be annotated to reach the same performance. Thus, using *pCT* is almost twice as fast as *Uncertainty* to reach a reasonable performance, while *multi_pCT* will shorten this process by half again. The benefits of better query selection are clearest for the first few batches of queries, which may be the range of greatest practical import for developers wanting to quickly add new event types.

Overall, we observe that pseudo co-testing performs better on all three evaluation metrics than uncertainty-based active learning. Uncertainty-based active learning requires more than 100 iterations before it reaches the level of

performance on all three measures achieved by the supervised system, trained on the entire corpus (*Full_Corpus)*. *pCT* takes 41 iterations to reach this level. At this point, there are in total 369+2050 = 2419 sentences in the training data; this represents a reduction in labor over sequential annotation of 80.6%. Applying the multi-criteria-based strategy (*multi_pCT*), we can reach this point even earlier, in iteration 23, where the labor is reduced by 87.8%[21].

# Chapter 7

# Conclusion

In this thesis, we analyzed the effect of wider scope features on event extraction. We mainly focused on using wider scope features to improve supervised event extraction systems. The first feature we explored is the supervised cross-event feature, which significantly improves the performance of a closed evaluation on ACE05 data. Then we explored the unsupervised topic features, which are especially useful when the testing data is a super-set of the training data, and the statistics from the training data is not accurate. Moreover, we

[21] We observe that the AL can perform better than training on the whole corpus; we believe that this is a result of AL selecting more positive training data. After 50 iterations of *multi-pcT*, 31.4% of the selected sentences have positive *Attack* examples, whereas only 6.1% of the entire corpus has such positive examples. Separate experiments suggest that using a corpus richer in positive examples can produce a small improvement in performance.

investigated how the wider scope can help semi-supervised and active learning approaches.

From the above studies, we can conclude that information from wider scope can aid event extraction based on local features, including different learning methods: supervised, semi-supervised, or active learning. Also, there are different ways to extract wider scope information from different levels, which need to be further explored. For example, can the different features be combined together, and which combination is the best? Can wider scope features help other NLP tasks, like relation extraction, named entity extraction, etc.?

# Bibliography

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In Proceedings of 5th ACM International Conference on Digital Libraries.

David Ahn. 2006. The stages of event extraction. In Proc. COLING/ACL 2006 Workshop on Annotating and Reasoning about Time and Events. Sydney, Australia.

Markus Becker and Miles Osborne 2005. A two-stage method for active learning of statistical grammars. In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence. Edinburgh, Scotland, UK.

David Blei, Andrew Ng, and Michael Jordan. 2003. Latent Dirichlet Allocation. Journal of Machine Learning Research 3: pp. 993–1022

K. Brinker. 2003. Incorporating Diversity in Active Learning with Support Vector Machines. In Proceedings of ICML, 2003.

Jinying Chen, Andrew Chein, Lyle Ungar and Martha Palmer. 2006. An empirical study of the behavior of active learning for word sense disambiguation. In Proceedings of the HLT-NAACL 2006. New York, USA.

D. Cohn, Atlas, L., & Ladner, R. 1994. Improving generalization with active learning. Machine Learning.

S. A. Engelson and I. Dagan. 1999. Committee- Based Sample Selection for Probabilistic Classifiers. Journal of Artifical Intelligence Research.

J. Finkel, T. Grenager, and C. Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In Proc. 43rd Annual Meeting of the Association for Computational Linguistics, pages 363–370, Ann Arbor, MI, June

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. In Proc. ACE 2005 Evaluation Workshop, Gaithersburg, MD.

Ralph Grishman. 2010. The impact of task and corpus on Event Extraction Systems. In Proceedings of LREC 2010

Prashant Gupta, Heng Ji. 2009. Predicting Unknown Time Arguments based on Cross-Event Propagation. In Proc. ACL-IJCNLP 2009

B. Hachey, B. Alex, and M Becker. 2005. Investigating the effects of selective sampling on the annotation task.. In proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005), 144-151. ACL, Ann Arbor, Michigan, USA.

Hilda Hardy, Vika Kanchakouskaya and Tomek Strzalkowski. 2006. Automatic Event Classification Using Surface Text Features. In Proc. AAAI06 Workshop on Event Extraction and Synthesis. Boston, MA.

Steven C.H Hoi, Rong Jin and Michael R. Lyu. 2006. Large-scale text categorization by batch mode active learning. Proceedings of the 15th International World Wide Web Conference (WWW 2006). Edinburgh, Scotland.

Heng Ji and R. Grishman. 2008. Refining Event Extraction through Cross-Document Inference. In Proc. ACL-08: HLT, pages 254–262, Columbus, OH, June.

R. Jones, R. Ghani, T. Mitchell, and E. Riloff. 2003. Active Learning for Information Extraction with Multiple View Feature Sets, ECML-03 Workshop on Adaptive Text Extraction and Mining

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-won Cha, and Gary Geunbae Lee. 2006. MMR-based active machine learning for bio named entity recognition. In Proceedings of the HLT-NAACL 2006. New York, USA.

Lewis and Gale 1994. A sequential algorithm for training text classifiers. In Proceedings of SIGIR '94, the 17th annual international ACM SIGIR conference on Research and development in information retrieval

Shasha Liao and Ralph Grishman. 2010a. Using Document Level Cross-Event Inference to Improve Event Extraction. In Proceedings of ACL 2010

Shasha Liao and Ralph Grishman. 2010b. Filtered Ranking for Bootstrapping in Event Extraction. In Proc. of COLING 2010.

Shasha Liao and Ralph Grishman. 2011a. Can Document Selection help Semi-supervised Learning? A Case Study on Event Extraction. In Proceedings of ACL 2011.

Shasha Liao and Ralph Grishman. 2011b. Acquiring Topic Features to Improve Event Extraction: in Pre-selected and Balanced Collections. In Proceedings of RANLP 2011.

Shasha Liao and Ralph Grishman. 2011c. Using Prediction from Sentential Scope to Build a Pseudo Co-Testing Learner for Event Extraction. In Proceedings of IJCNLP 2011.

Ting Liu. 2009. Bootstrapping events and relations from text. Ph.D. thesis, State University of New York at Albany.

Gideon Mann. 2007. Multi-document Relationship Fusion via Constraints on Probabilistic Databases. In Proceedings of HLT/NAACL 2007. Rochester, NY, US.

M. Maslennikov and T. Chua. 2007. A Multi resolution Framework for Information Extraction from Free Text. In Proc. 45th Annual Meeting of the Association of Computational Linguistics, pages 592–599, Prague, Czech Republic, June.

A. McCallum and K. Nigam. 1998. Employing EM in Pool-Based Active Learning for Text Classification. In Proceedings of ICML, 1998.

MUC-4 Proceedings. 1992. Proceedings of the Fourth Message Understanding Conference (MUC-4). Morgan Kaufmann

MUC. 1995. Proceedings of the Sixth Message Understanding Conference (MUC-6), San Mateo, CA. Morgan Kaufmann.

I. Muslea, Minton, S., & Knoblock, C. (2000) Selective sampling with redundant views. Proc. Of National Conference on Artificial Intelligence (pp. 621-626)

Miles Osborne and Jason Baldridge. 2004. Ensemble-based active learning for parse selection. In Proceedings of Human Language Technology Conference- the North American Chapter of the Association for Computational Linguistics Annual Meeting. (HLT-NAACL 2004). Boston, Massachusetts, USA.

S. Patwardhan and E. Riloff. 2007. Effective Information Extraction with Semantic Affinity Patterns and Relevant Regions. In Proc. Joint Conference on Empirical

Methods in Natural Language Processing and Computational Natural Language Learning, 2007, pages 717–727, Prague, Czech Republic, June.

S Patwardhan and E. Riloff. 2009. A Unified Model of Phrasal and Sentential Evidence for Information Extraction. In Proc. Conference on Empirical Methods in Natural Language Processing 2009, (EMNLP-09).

http://projects.ldc.upenn.edu/ace/docs/English-Events-Guidelines_v5.4.3.pdf

Daniel Ramage, David Hall, Ramesh Nallapati, Christopher D. Manning 2009. Labeled LDA: a supervised topic model for credit attribution in multi-labeled corpora. In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing.

Roi Reichart and Ari Rappoport 2007. An ensemble method for selection of high quality parses. In Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL 2007).

Ellen Riloff. 1996. Automatically Generating Extraction Patterns from Untagged Text. In Proc. Thirteenth National Conference on Artificial Intelligence (AAAI-96), 1996, pp. 1044-1049.

Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi and Deryle Lonsdale 2007. Active Learning for part-of-speech tagging: Accelerating corpus annotation. In proceedings of the Linguistic Annotation Workshop. ACL, Prague, Czech Republic. 2007.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000). Stanford, California, USA.

D Shen, J Zhang, J Su, and G Zhou. 2004. Multi-Criteria-based Active Learning for Named Entity Recognition. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, 2004.

Trevor Strohman, Donald Metzler, Howard Turtle and W. Bruce Croft. 2005. Indri: A Language-model based Search Engine for Complex Queries (extended version). Technical Report IR-407, CIIR, UMass Amherst, US.

Mihai Surdeanu, Jordi Turmo, and Alicia Ageno. 2006. A Hybrid Approach for the Acquisition of Information Extraction Patterns. Proceedings of the EACL 2006 Workshop on Adaptive Text Extraction and Mining (ATEM 2006)

M. Stevenson and M. Greenwood. 2005. A Semantic Approach to IE Pattern Induction. In Proceedings of ACL 2005.

M. Tang, X. Luo and S. Roukos. 2002. Active Learning for Statistical Natural Language Parsing. In Proceedings of the ACL 2002.

C. A. Thompson, M. E. Califf and R. J. Mooney. 1999. Active Learning for Natural Language Parsing and Information Extraction. In Proceedings of ICML 1999.

Simon Tong and Daphne Koller 2002. Support vector machine active learning with applications to text classification. Journal of Machine Learning 2 (Match): 45-66

S. Tong, and D. Koller. 2001a. Active learning for parameter estimation in Bayesian networks. Advances in Neural Information Processing Systems 13 (pp. 647–653).

S. Tong, and D. Koller. 2001b. Active learning for structure in Bayesian networks. Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (pp. 863–869).

Roman Yangarber, Ralph Grishman; Pasi Tapanainen; Silja Huttunen. 2000. Automatic Acquisition of Domain Knowledge for Information Extraction. In Proc. COLING 2000.

Roman Yangarber. 2003. Counter-Training in Discovery of Semantic Patterns. In Proceedings of ACL2003.

Roman Yangarber and Lauri Jokipii. 2005. Redundancy-based Correction of Automatically Extracted Facts. In Proceedings of HLT/EMNLP 2005. Vancouver, Canada.

Roman Yangarber. 2006. Verification of Facts across Document Boundaries. In Proceedings of International Workshop on Intelligent Information Access. Helsinki, Finland.

Roman Yangarber, Clive Best, Peter von Etter, Flavio Fuart, David Horby and Ralf Steinberger. 2007. Combining Information about Epidemic Threats from Multiple Sources. In Proceedings of RANLP 2007 workshop on Multi-source, Multilingual Information Extractionand Summarization. Borovets, Bulgaria.

David Yarowsky. 1995. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In Proceedings of ACL 1995. Cambridge, MA.

Li Zhang, Yue Pan, Tong Zhang. 2004. Focused named entity recognition using machine learning. In Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval.

Jingbo Zhu and Eduard Hovy 2007. Active Learning for word sense disambiguation with methods for addressing the class imbalance problem. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning.

Xiaojin Zhu. 2008 Semi-Supervised Learning Literature Survey. http://pages.cs.wisc.edu/~jerryzhu/research/ssl/semireview.html