Event Extraction for Document-Level Structured Summarization

Andrew Hsi

June 2017

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Yiming Yang, Co-Chair
Jaime Carbonell, Co-Chair
Alexander Hauptmann
Michael Mauldin

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Event Extraction, Summarization, Structured Prediction, Deep Learning

Abstract

Event extraction has been well studied for more than two decades, primarily through the lens of the Message Understanding Conferences (MUC) and Automatic Content Extraction (ACE) programs. However, event extraction methods to date do not yet offer a satisfactory solution to providing concise, structured, document-level summaries of events in news articles. Prior work in ACE focuses on fine-grained sentence-level events, which do not offer good document-level summaries of events. Previous work under MUC relied heavily on handcrafted rules for highly specific domains, resulting in models that are not easily generalizable to new domains.

In this thesis, we propose a new framework for extracting document-level event summaries called *macro-events*, unifying together aspects of both information extraction and text summarization. The goal of this work is to extract concise, structured representations of documents that can clearly outline the main event of interest and all the necessary argument fillers to describe the event. Unlike work in abstractive and extractive summarization, we seek to create template-based, structured summaries, rather than plain text summaries.

We propose two novel methods to address this problem. First, we introduce a structured prediction model based on the Learning to Search framework for jointly learning argument fillers both across and within event argument slots. Second, we propose a deep neural model that treats the problem as machine comprehension, which does not require training with any on-domain macro-event labeled data. Our initial experiments on filling macro-event templates for two domains (attacks and elections) show strong performance under both models compared to existing baselines.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Thesis Statement and Contributions

The problem of event extraction seeks to identify instances of events in texts, along with any arguments corresponding to the roles in the event. Such a task has obvious real-world applicability, as it allows users to quickly identify the who, what, where, and when of news stories without having to manually read through the text. Moreover, it provides structured output for additional analytic processes such as cross-document coreference, script-induction, or other higher level text mining tasks. A key problem with current event extraction methods is that the granularity produced by systems does not match the granularity desired by users. The first mention of an event may identify the "what" and "when", but the "who" and "where" (or other roles) may come in later mentions. Ideally, a comprehensive event summary should be presented to a user in a concise, structured form, centered around the main event described in the text. For example, the output of event extraction on a news story about a shooting could identify the people involved, the names of anyone who was injured or killed, where the shooting occurred, who the suspects may be, and when the shooting occurred. Table 1.1 presents an example document-level summary of a single news article about a shooting.

Existing frameworks for events fall under two different paradigms. Most recent work follows the standards set by the Automatic Content Extraction (ACE) program[1] [11, 29, 38, 48, 49, 51, 62, 82]. ACE events focus on sentence-level events rather than a document-level focus, and while the event types considered are general enough to be of interest to common users, they are too fine-grained for summarizing a document's content. Any particular document could have an arbitrary number of events, of varying importance to the overall document content (see Table 1.2 for an example of ACE-style output on the same text summarized in Table 1.1). Events studied in the Message Understanding Conferences[2] (MUC) [14, 15, 27, 28, 36, 37, 50, 64, 71, 76, 83], held from 1987 to 1997, had only one template per document, and are thus closer to the desired granularity, but relied heavily on hand-crafted rules and focused only on a small set of very specific event templates (e.g. terrorist events in Latin America, electronic circuit fabrication, and launches of rockets/missiles), which are not generalizable enough to cover a wide set of common

[1] http://www.itl.nist.gov/iad/mig/tests/ace/
[2] http://www-nlpir.nist.gov/related_projects/muc/

public interests.

In this thesis, we propose a new event extraction task for extracting document-level structured summaries, called *macro-events*. Our assumption is that for many documents the text can be represented by a single macro-event template, which focuses on the dominating event of the text. We constrain the set of defined argument fillers for each event type to a predefined, but highly interdependent set of categories containing key event information. This allows for events to be represented in a concise, yet highly informative manner to users, and to further processing by downstream analytical methods. Finally, we seek to keep the notion of a macro-event generalizable across a wide set of event types, which is key to developing event structures that have broad applicability and that can be useful beyond toy examples.

In principle, perfectly solving ACE-style event extraction could enable subsequent perfect solutions to the macro-event task. However, to do so would require near-perfect sentence-level event extraction, near-perfect event coreference [1, 44, 56], near-perfect entity coreference [44, 45], near-perfect aggregation of sentence-level events to a document-level template, and near perfect information extraction across entities, event triggers, and event argument, so as to mitigate against compounding errors in the processing pipeline. Output for existing ACE-style extraction algorithms come with a multitude of problems, including redundancy, incorrect extractions, off-topic extractions, contradictory extractions, and empty events devoid of argument fillers (see Table 1.2 for examples of these problems). We argue that it is instead better to focus directly on the macro-event task itself, which does not require near-perfectly solving each and every subproblem.

To solve the macro-event extraction problem means addressing the following points. First, arguments within an event have complex dependencies that must be modeled when attempting to fill up a template. For example, knowledge about the location and time of an attack may be needed to disambiguate between the victims of two separate shootings. Second, any approach for event extraction that jointly models such argument relationships must be able to handle the enormous output space that results from considering all possible combinations of argument fillers. Finally, we must ensure that our techniques are able to effectively learn models even in a data-scarce scenario, as annotated training event corpora rarely exceed more than a few hundred documents in size. We introduce a novel approach based on Learning to Search, a general machine learning framework for structured prediction [19] that addresses all three of these problems, achieving high performance on the macro-event extraction task. We further introduce a new approach based on neural machine reading comprehension that can achieve high performance over baseline methods even when little to no training data exists for the target domain. To the best of our knowledge, this is the first time that deep neural networks for machine reading have been applied to the event extraction domain.

Outside of the MUC and ACE notions of events, other related approaches for summarizing a text include topic models, extractive summarization, and abstractive summarization. Yet none of these can fulfill the needs of the macro-event extraction task. Topic models like LDA [2] have latent factors that are difficult to interpret, and do not offer insight into the fillers of arguments. Both extractive [3, 4, 17, 21, 52, 53, 61, 66, 79] and abstractive summarization [24, 25, 26, 40, 54, 65, 74, 75] can only be used to compress or rewrite the text into a shorter form – neither can create a structured output of the form seen in Table 1.1 that is useful both for human readers and down-stream analytic processes.

Attack Macro-Event	
Perpetrator	Jason Brian Dalton
Victims – Dead	Richard Smith Tyler Dorothy Brown Barbara Hawthorne Mary Lou Nye Mary Jo Nye
Victims – Injured	Tiana Carruthers Abigail Kopf
Time	Saturday
Location	Kalamazoo

Table 1.1: An ideal output for summarizing documents via events. The main point of the text is clear, and fillers for each role are easily identifiable.

1.2 Completed Work

To date, we have already completed a sizable amount of work on the task of macro-event extraction:

- Formulation of the task of macro-event extraction for structured summarization of documents. The macro-event structure represents the unification of event ideologies studied in the ACE and MUC programs, focusing on single event per document summaries (like MUC) that are generalizable enough to cover a wide range of event types (like ACE).
- Development of a novel approach for structured learning of macro-event templates using the Learning to Search framework that jointly learns the fillers both within and across argument roles.
- Adaptation of a state-of-the-art neural machine reading comprehension algorithm to the task of extracting macro-event templates. The model is trained entirely on off-domain data, and can perform macro-event extraction for domains with zero available labeled macro-event data.
- Collection and annotation of initial macro-event gold standard data for the *attack* and *election* domains. Labels have been obtained using a combination of manual annotation and infobox extraction from Wikipedia articles.
- Preliminary experimental results on the *attack* and *election* domains with baseline methods and both of our proposed algorithms.

1.3 Proposed Work

Our proposed work for completion of this thesis is as follows:

- Expansion of the annotated macro-event corpus to additional domains. Our current corpus contains annotated macro-events for the attack and election domains. We seek to expand

annotation to an additional two macro-event types: sporting events and criminal trials. (Ongoing work)

- Modification of our neural model to utilize annotated macro-event training data. Our current model shows strong performance in settings where little on-domain training data exists, but does not have any capability to take advantage of scenarios where such data does exist. We propose to explore methods for adapting our neural model to the target domain using gold-standard annotated training data for macro-events.
- Extension of our Learning to Search model to handle multilingual input. Although our current model is designed to work on English texts, the algorithm is adaptable to other languages. To demonstrate this generalizability, we will augment this model to handle one additional language.
- Full experiments on the complete macro-event English annotated corpus. We will apply our proposed methods and baselines to all four annotated macro-event domains: attacks, elections, sporting events, and criminal trials.
- Multilingual experiments for macro-event extraction. We will expand annotation of one macro-event domain to a non-English language, and run an additional set of experiments on this data to demonstrate our model's ability to handle non-English texts.

1.4 Proposal Outline

The rest of this thesis proposal is organized as follows. Chapter 2 describes related work on event extraction, machine reading, and summarization. Chapter 3 describes the macro-event framework in more detail. In Chapter 4, we introduce our proposed structured model for filling up macro-events, and describe connections between this approach and policy gradients. In Chapter 5, we introduce our second model, based on recent advances in neural machine reading techniques. Chapter 6 describes our preliminary experimental results on filling macro-events from text. We present our planned timeline of work in Chapter 7 and conclude in Chapter 8.

Event Type	**Argument Role**	**Argument Fillers**
Justice.Charge-Indict	Defendant	Kalamazoo shooting suspect Jason Brian Dalton
Justice.Charge-Indict	Defendant	CNN)Authorities
Life.Die		
Movement.Transport	Argument	the shooter
Conflict.Attack		
Life.Die	Victim	six people
Conflict.Attack		
Conflict.Attack	Attacker Target	the gunman eight people
Conflict.Attack	Target	a woman
Life.Die	Agent Victim	the gunman a father and son
Movement.Transport	Artifact Destination	he a Cracker Barrel restaurant
Life.Die	Place Victim Victim	a Cracker Barrel restaurant four women a 14-year-old girl
Life.Injure	Victim	a 14-year-old girl
Conflict.Attack		
Life.Die		
Justice.Jail	Agent Person Place	police Dalton, 45 downtown Kalamazoo
Transaction.Transfer-Ownership	Buyer Artifact	Police a weapon
...	...	...
Conflict.Attack	Target	Tiana Carruthers
Conflict.Attack	Target Instrument	both a vehicle
Life.Die	Victim	four women
Conflict.Attack	Target	a 14-year-old girl
Life.Die	Victim	many more victims
Life.Die	Victim	more people
Justice.Trial-Hearing		

Table 1.2: Partial output from running a sentence-level event extraction system. The complete version has 41 distinct events extracted from the text. Many of the extracted events are redundant (e.g. "four women" appears repeatedly as a Life.Die argument), devoid of attached arguments, incorrect (e.g. "CNN)Authorities" is not a trial defendant), contradictory (e.g. "a 14-year-old girl" is a victim of both injure and kill events), or simply off-topic to the main point of the document (e.g. Transaction.Transfer-Ownership is not important enough for a summary).

Chapter 2

Related Work

2.1 Event Extraction

Event extraction dates back to 1987 with the DARPA MUC program. Each year of the MUC program focused on a single type of event template, allowing for the study of complex structure and fillers within the event template (see Table 2.1 for a complete list of event types considered). However, this limits the generalizability of the resulting models, as the training data and event templates for each year are tailored to very specific scenarios, and thus can only be reused on a very narrow range of texts. Furthermore, the complexity of the resulting templates is undesirable for common users, who may only care about seeing the most important argument fillers in the text. Methods studied to approach the MUC-style event task fall into two main categories: pattern-matching based algorithms [27, 36, 37, 50, 71, 76, 83] and classification-based algorithms [14, 64].

Year	Event type
1987	Fleet Operations
1989	Fleet Operations
1991	Terrorist activities in Latin America
1992	Terrorist activities in Latin America
1993	Corporate Joint Ventures, Microelectronic production
1995	Negotiation of Labor Disputes and Corporate Management Succession
1997	Airplane crashes, and Rocket/Missile Launches

Table 2.1: Event types studied in the MUC program by year

Beginning with the ACE program, researchers have switched focus to more general types of events, such as conflicts, transportation of people/items, and life events (see Table 2.2 for a complete list of events considered). This allows the program to capture a much wider range of event types than the MUC program. However, the scope of an event was reduced from document level to sentence level. For each triggered event (indicated by some word in the text, usually a predicate or noun), the scope of arguments is limited to only the rest of the sentence. Furthermore, every extracted event produced by systems is treated with equal importance, so there is no

meaningful way to determine what the "main" focus of the document was. Traditional methods for ACE extraction focus on pipelines of classifiers [29, 38, 51], with some more recent work considering structured models [48, 49] and neural approaches [11, 62].

There are some cases in which event extraction algorithms have been deployed to additional languages beyond English. The MUC-5 conference considered event templates in Japanese for both the joint ventures and micro-electronics domains [36, 37]. ACE-style event extraction has been applied to Chinese and Spanish [8, 9, 12, 13, 33, 34, 35, 46]. However, the majority of event extraction work focuses exclusively on English texts.

In addition to textual event extraction, there exists a large body of work applying event extraction to video [55, 57, 58, 60, 68, 69, 77, 80, 81]. While video event extraction is outside the scope of this thesis, we believe macro-event extraction techniques could prove useful for creating template-based summaries of multimedia. Just as in texts, videos also have the problem of identifying 1.) the main event, 2.) the participants in the event, and 3.) the various argument roles to which each participant belongs. For example, consider the video feed from a security camera. While most of the video may be unimportant, it is highly desirable to be able to concisely summarize any content containing a major event, such as an attack or robbery. In order to provide a complete video summary, it is necessary to identify who is involved in the event, and what roles they fall under (e.g. attacker, victim, police) – all of which closely parallels the content seen in macro-event templates for texts.

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

Table 2.2: Event types and subtypes considered in the ACE program

2.2 Machine Reading Comprehension

The field of machine reading comprehension seeks to create algorithms that can read, understand, and reason about texts. Machine reading comprehension can be seen as a specific case of the question answering problem, where the available knowledge to the system is restricted to a single source (e.g. a single document or passage), rather than having access to the entire web. A system that is able to succeed on such a task should be in principle well suited to event extraction (and more broadly, information extraction), as any system that can reason about documents should be able to also extract relevant entities, relations, and events from the texts.

Machine text comprehension is not a new problem. For decades, researchers have attempted to develop models that can read a text and answer questions about its content [6, 7, 22, 32, 70, 70, 72]. However, these systems were severely limited by the amount of available annotated data – typically only in the hundreds of examples.

The field has recently exploded in popularity due to advances in deep learning and recent development of massive corpora for machine reading comprehension. Hermann et al. [30] introduced the CNN/Dailymail corpora – two massive datasets generated automatically by harvesting article summaries from online news stories. These datasets contain hundreds of thousands of examples, and are vastly larger than any other machine reading comprehension datasets created before then. The experiments of Hermann et al. on this data with deep learning models showed vastly superior performance compared to non-neural models. Since then, numerous advances have been made, both in terms of additional available datasets [31, 63, 67] and more sophisticated deep learning models [10, 18, 20, 41].

2.3 Summarization

The goals of summarization closely match those of our proposed macro-events, as both tasks seek to provide a high-level, concise view of a document. Summarization methods fall into two categories: abstractive and extractive summarization. Abstractive summarization focuses on generating natural language summaries that can describe the document in a shorter or simpler form [24, 25, 26, 40, 54, 65, 74, 75]. Extractive summarization avoids the problem of natural language generation by instead extracting subsets of the original text in order to create a summary [3, 4, 17, 21, 52, 53, 61, 66, 79].

Ultimately however, neither abstractive nor extraction summarization meets the goal of our problem, as both methods simply produce an unstructured natural language text. In contrast, our proposed framework does not seek to generate natural language summaries, but instead template-based structured summaries, which we believe can offer quicker access to the information needs of users and more importantly, feed downstream automated analytic capabilities.

Some works on text summarization have considered how to incorporate events into summaries. Filatova and Hatzivassiloglou [23] extracted relationships between entities and frequent nouns to represent events, and used these as features for extractive summarization. Ji et al. [39] explored using an information extraction system to identify entities, relations, and events, and utilize these to reweight candidate sentences in an extractive summarization system. However, these kinds of studies are ultimately still considering the task of generating natural language summaries, rather than template-based summaries.

Chapter 3

Macro-Events

3.1 Definition

In this chapter, we introduce the concept of "macro-events". A macro-event is a structured template providing a summarization of a single document through argument slot filling. The goal of such a template is to provide a high-level view of a document's content, focused solely on the primary event described in the text.

Like in information extraction, we formulate our objective as the problem of filling up a predefined template with particular strings from the text. This can be broken down into three key subproblems: 1.) determining the primary event of interest in the document, 2.) classifying the event type into one of several predefined types, and 3.) extracting entities associated with each argument slot. This is notably different from the paradigms studied in MUC and ACE. In MUC, the event type for each year was always given, so there was no need to determine the correct template to use for any particular document. In ACE, it was necessary to determine the type of event template to use for each template, but participants did not have to distinguish between the main event of the document and any secondary events that might also be mentioned in the text.

More formally, a macro-event takes the form of the template seen in Table 3.1. It consists of a macro-event type, a set of argument roles, and a set of named entity fillers for each of these roles.

Macro-Event Type	
Argument Type 1	Named entity fillers
Argument Type 2	Named entity fillers
...	...
Argument Type N	Named entity fillers

Table 3.1: A macro-event template prototype. A macro-event consists of a type, argument roles, and argument fillers. Each of the argument fields in a completed macro-event may be filled by zero, one, or more textual fillers.

The type of the macro-event template is simply the ontological class to which the event belongs. This corresponds exactly to the notion of an event type in ACE. Example event types

include *attack*, *natural disaster*, *transaction*, *election*, or *demonstration*.

The argument roles of the macro-event template are the specific roles that define an event of this type. For instance, if considering an *election* event, the event could be well-defined by roles of *location*, *time*, *position*, *nominees*, *winner*, and *losers*.

For some events, it can be desirable to organize the argument roles into a hierarchy. For instance, continuing the *election* example, we can define a two-layer macro-event template, where the first layer includes the roles *location*, *time*, *position*, and *nominees*, and the second layer includes the roles of *winner* and *losers* as children of the *nominees* role. See Figure 3.1 for a visualization of this example, and Figures 3.2 and 3.3 for an examples of the *attack* and *transaction* template structure.

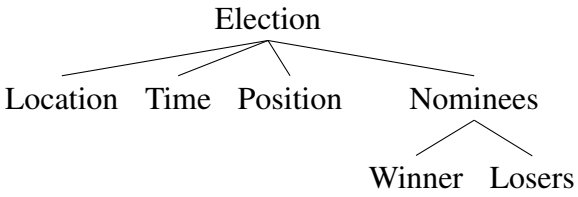


Figure 3.1: Hierarchy defined by the *election* macro-event type

The hierarchy structure allows for easy definition of constraints among the entity fillers. Sibling relationships between roles in the hierarchy imply cannot-link relationships. Thus, in the *election* macro-event, any filler that is used in the *Location* slot cannot also be used in the *Position* slot.

The parent-child relationship is also meaningful. Any filler of a child role must also be a filler of a parent role. In the *election* macro-event, this means that any winner or loser of an election must also be a nominee for the election.

Overall, this kind of hierarchical structure provides a very simple and clean way to define any sorts of constraints that arguments may be subject to within an event of interest.

Each argument filler corresponds to a specific span of text from the original document. In general, an arbitrary number of entities could fill any particular role, although in practice some may typically be filled by just a single text span. For example, *time* and *location* for an election typically have just a single value, while *nominees* will in most cases require multiple fillers.

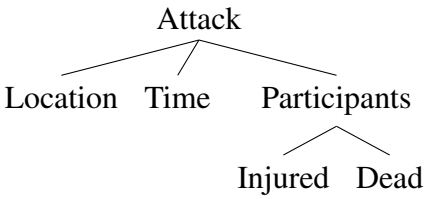


Figure 3.2: Hierarchy defined by the *attack* macro-event type

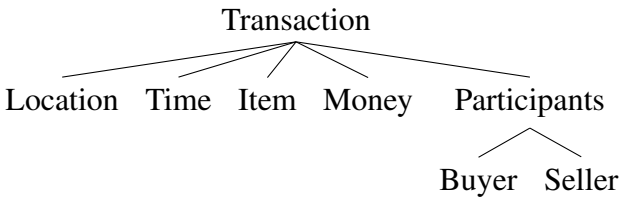


Figure 3.3: Hierarchy defined by the *transaction* macro-event type

3.2 Event Ontology

In this thesis, we propose to work on multiple event types, in order to demonstrate the generalizability of the macro-event paradigm and our algorithms to a variety of domains. In this section, we will describe our planned domains and their current macro-event templates according to our in-progress work. We will describe the actual annotation process in detail in the following section.

3.2.1 Attacks

The *attack* macro-event consists of the following argument roles:

- Location – where the attack took place. In annotation, we prioritize city name first; if this does not exist then state, then country, then any applicable named entity.
- Time – when the attack took place. In annotation, we mark the most informative single span of text in the document.
- Participants – who was involved in the attack. This includes attackers, targets, and victims.
- Injured – anyone who was injured in the attack, but not killed.
- Dead – anyone who died from the attack.

The hierarchical structure for attack can be seen in Figure 3.2.

3.2.2 Elections

The *election* macro-event consists of the following argument roles:

- Location – where the election took place.
- Time – when the election took place.
- Title – the position the election was for (e.g. President, Senator)
- Nominees – the candidates nominated for the election
- Winner – the candidate who won the election
- Losers – the candidates who lost the election

The hierarchical structure for attack can be seen in Figure 3.1.

3.2.3 Sporting Events (Proposed Work)

The *sporting-event* macro-event consists of the following argument roles:

- Location – where the sporting event took place
- Time – when the sporting event took place
- Type of Sport – the particular type of sport in the event
- Competitors – the people and/or teams participating in the event
- Winner – the people and/or team that won the event
- Losers – the people and/or teams that lost the event

The hierarchical structure for sporting events can be seen in Figure 3.4.

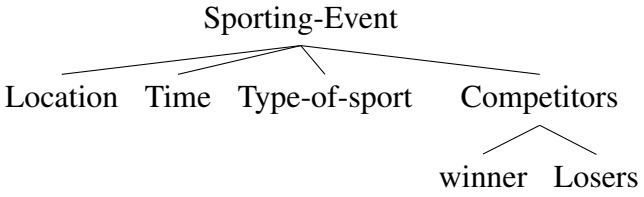


Figure 3.4: Hierarchy defined by the *sporting-event* macro-event type

3.2.4 Criminal Trials (Proposed Work)

The *criminal-trial* macro-event consists of the following argument roles:

- Location – where the trial took place
- Time – when the trial took place
- Crime – the type of crime the trial is about
- Defendant – the person under trial
- Verdict – whether the defendant won or lost the trial
- Sentence – the punishment (if any) that the defendant must serve

The hierarchical structure for sporting events can be seen in Figure 3.5.

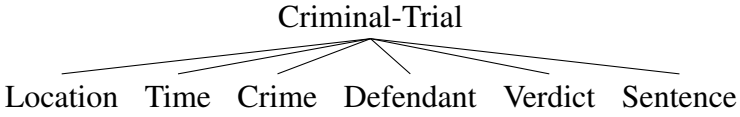


Figure 3.5: Hierarchy defined by the *criminal-trial* macro-event type

3.3 Annotation

In this section, we will describe the process by which we annotate documents. We will be using several techniques for obtaining gold-standard annotation, dependent on the domain and

availability of existing resources.

3.3.1 Internal Annotation

For domains without sufficiently detailed existing resources, we will be conducting internal annotation of documents. Our annotation process is as follows. For the purposes of illustrating our process, we will use the *attack* domain as a running example.

For each document, the annotator first establishes what the main event of the document is, as well as its corresponding event type. In some documents, this may be simple – for instance, a document which describes a single attack, and no other events. In other documents, determining the main focus requires more effect. For example, documents about shootings may mention another, separate *attack* event for the purpose of providing historical context – such additional events are not considered during annotation.

Once the main event is established, the annotator identifies named entities in the document and assigns them to any relevant argument roles in the macro-event template. The annotator is not required to explicitly mark every named entity in the text; only those that actually participant in the main event as a filler.

An important concern in annotating fillers is that many entities have multiple forms used within the same document (e.g. "Obama", "Barack Obama", "Barack Hussein Obama"). In such cases, the annotator only marks the most complete, canonical form of the entity in question. For example, if a document mentions both "Barack Obama" and "Obama", the annotator would only fill in "Barack Obama", which is the more complete form of his name.

During annotation, some rules based on the hierarchical structure may automatically be applied to ease the process. For instance, when annotating attack macro-events, any fillers marked under the "injured" slot may automatically be marked under "participants" as well.

To date, we have already conducted annotation of documents for the *attack* domain. We collected documents from CNN.com, with publication dates spanning the years 2014-2016. Corpus statistics may be seen in Table 3.2. We expect to use this kind of annotation for the *criminal-trial* domain as well.

Corpus Statistics		
	Attack	Election
Total # of documents	242	1420
Average words per document	697.85	481.85
Maximum words per document	2604	16930
Average fillers per document	5.16	7.26
Maximum fillers per document	35	21
Average fillers per slot	1.03	1.21
Maximum fillers per slot	16	9

Table 3.2: Statistics for the *Attack* and *Election* macro-event corpora

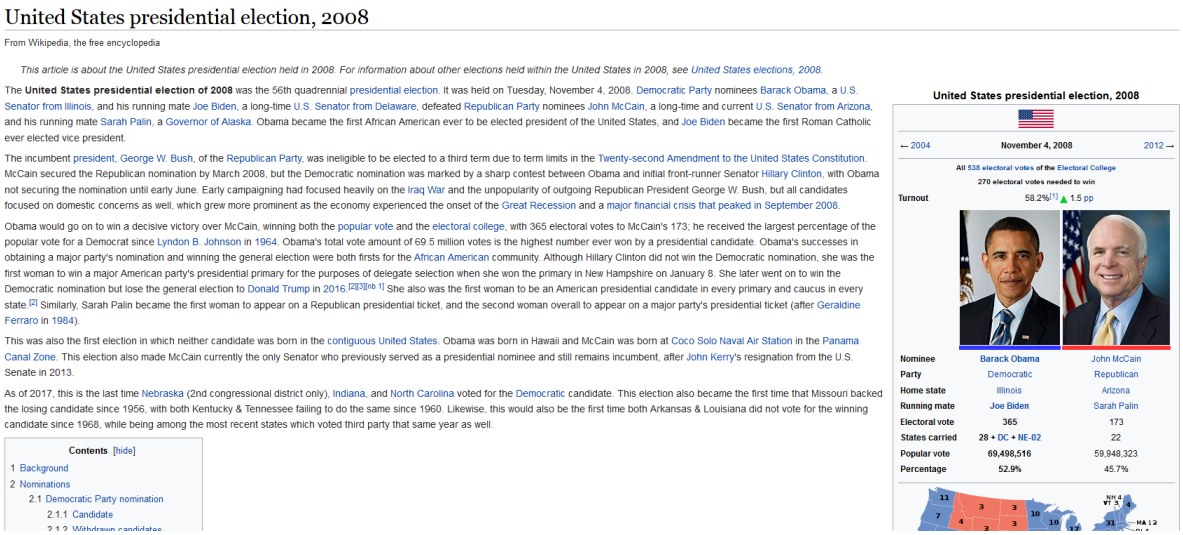
Although most of our work with document annotation is likely to be limited to a small number of documents, we are planning to do larger-scale annotation for at least one domain. We plan to

achieve this by hiring annotators, either via local students or Amazon Mechanical Turk.

3.3.2 Leveraging Online Resources for Annotation

For some domains, it is possible to leverage existing resources for use as gold-standard information. In particular, we have looked at Wikipedia infoboxes, which are a rich resource containing key information for their respective documents.

Wikipedia infoboxes are template-based structures containing slots and fillers, typically used to summarize key entities and relationships within an article. For example, the Wikipedia infobox associated with the 2008 United States Presidential Election (see Figure 3.6) contains information about the presidential candidates, the vice-presidential candidates, their political parties, the number of electoral votes received, and so on. In 2010, it was estimated that approximately one third of Wikipedia articles contained infoboxes [43].


United States presidential election, 2008

From Wikipedia, the free encyclopedia

This article is about the United States presidential election held in 2008. For information about other elections held within the United States in 2008, see United States elections, 2008.

The **United States presidential election of 2008** was the 56th quadrennial presidential election. It was held on Tuesday, November 4, 2008. Democratic Party nominees Barack Obama, a U.S. Senator from Illinois, and his running mate Joe Biden, a long-time U.S. Senator from Delaware, defeated Republican Party nominees John McCain, a long-time and current U.S. Senator from Arizona, and his running mate Sarah Palin, a Governor of Alaska. Obama became the first African American ever to be elected president of the United States, and Joe Biden became the first Roman Catholic ever elected vice president.

The incumbent president, George W. Bush, of the Republican Party, was ineligible to be elected to a third term due to term limits in the Twenty-second Amendment to the United States Constitution. McCain secured the Republican nomination by March 2008, but the Democratic nomination was marked by a sharp contest between Obama and initial front-runner Senator Hillary Clinton, with Obama not securing the nomination until early June. Early campaigning had focused heavily on the Iraq War and the unpopularity of outgoing Republican President George W. Bush, but all candidates focused on domestic concerns as well, which grew more prominent as the economy experienced the onset of the Great Recession and a major financial crisis that peaked in September 2008.

Obama would go on to win a decisive victory over McCain, winning both the popular vote and the electoral college, with 365 electoral votes to McCain's 173; he received the largest percentage of the popular vote for a Democrat since Lyndon B. Johnson in 1964. Obama's total vote amount of 69.5 million votes is the highest number ever won by a presidential candidate. Obama's successes in obtaining a major party's nomination and winning the general election were both firsts for the African American community. Although Hillary Clinton did not win the Democratic nomination, she was the first woman to win a major American party's presidential primary for the purposes of delegate selection when she won the primary in New Hampshire on January 8. She later went on to win the Democratic nomination but lose the general election to Donald Trump in 2016.[2][3][nb 1] She also was the first woman to be an American presidential candidate in every primary and caucus in every state.[2] Similarly, Sarah Palin became the first woman to appear on a Republican presidential ticket, and the second woman overall to appear on a major party's presidential ticket (after Geraldine Ferraro in 1984).

This was also the first election in which neither candidate was born in the contiguous United States. Obama was born in Hawaii and McCain was born at Coco Solo Naval Air Station in the Panama Canal Zone. This election also made McCain currently the only Senator who previously served as a presidential nominee and still remains incumbent, after John Kerry's resignation from the U.S. Senate in 2013.

As of 2017, this is the last time Nebraska (2nd congressional district only), Indiana, and North Carolina voted for the Democratic candidate. This election also became the first time that Missouri backed the losing candidate since 1956, with both Kentucky & Tennessee failing to do the same since 1960. Likewise, this would also be the first time both Arkansas & Louisiana did not vote for the winning candidate since 1968, while being among the most recent states which voted third party that same year as well.

Contents [hide]
1 Background
2 Nominations
2.1 Democratic Party nomination
2.1.1 Candidate
2.1.2 Withdrawn candidates

United States presidential election, 2008

← 2004 November 4, 2008 2012 →

All 538 electoral votes of the Electoral College
270 electoral votes needed to win

Turnout 58.2%[1] ▲ 1.5 pp

Nominee	Barack Obama	John McCain
Party	Democratic	Republican
Home state	Illinois	Arizona
Running mate	Joe Biden	Sarah Palin
Electoral vote	365	173
States carried	28 + DC + NE-02	22
Popular vote	69,498,516	59,948,323
Percentage	52.9%	45.7%

Figure 3.6: Example Wikipedia infobox and its corresponding document

For certain domains, the coverage of infobox template fields is sufficiently broad to contain all of the information needed to fill macro-event templates. In such cases, gold standard annotations for documents can be obtained by simply parsing their infoboxes (see Table 3.3 for an example macro-event template extracted from an infobox). We have created an election-based macro-event corpus using this approach, and are planning to repeat this process for the *sporting-event* domain.

Election Macro-Event	
Title	President
Nominees	Barack Obama John McCain
Winner	Barack Obama
Losers	John McCain
Time	November 4, 2008
Location	United States

Table 3.3: A gold-standard macro-event template extracted from the infobox seen in Figure 3.6.

Chapter 4

Learning to Search

Intuitively, jointly learning all of the fillers for each slot in the template is the optimal way to solve this problem, as it provides the most constraints and dependencies. Knowing the name of one person involved in an attack should make it easier to identify other people who were also involved. Similarly, knowing the position field in an election should make it easier to identify the nominees. In general, using global information about the rest of the template in this way is a more informative way of filling up a template than by simply making independent decisions.

However, a naive approach to joint inference is computationally prohibitive. Suppose we wish to fill up just a single slot in a template, and are given n entity candidates that could be used to fill in the slot. The true answer could be that all of the candidates are fillers, that none of them are fillers, or any subset of them are fillers. Thus, to consider even just a single slot means considering the power set of entity candidates: 2^n possible combinations. Further expanding this problem to fill in *all* of the slots in a template becomes even more expensive.

4.1 Our Proposed Model

In order to leverage global information, while maintaining computational intractability, we adopt the Learning to Search framework for structured prediction. The main idea of Learning to Search is to reframe the problem of structured prediction as a reinforcement learning problem [42]. Let us begin by defining some key terminology. Formally, a policy π is a mapping from any state s_t (represented by document-related features plus historical decisions) at step t to the action a_t of filling a particular slot with a specific entity candidate. In practice under this framework, the policy is simply represented by a classifier that determines the correct action to take (among a fixed set of possible choices), conditioned on the current state. The system will transit to its next state after taking each action. To measure the effectiveness of π, a reward will be triggered at each action a_t if the action results in the inclusion of a correct argument filler. Our overall goal is to learn a policy that maximizes the rewards of the system. We can improve beyond our initial policy by allowing the system to explore the data. The main benefit of exploring the data is that it allows the system to be more robust at test time to classification paths unseen in the original training data.

More specifically, we follow the AggreVaTe model of Ross and Bagnell [73], and apply it to

the task of macro-event extraction as follows. For each document, we start by collecting all entity candidates (via named entity recognition), and pairing each one with all possible argument roles in the macro-event structure (e.g. $(time, Sunday)$, $(injured, JohnDoe)$, $(location, Boston)$ for an *attack* macro-event). For each of these pairs, we need to decide to either include or exclude the pair from the final event template. These decisions can be made independently, but doing so may result in violating the constraints defined by our macro-event template hierarchy. Instead, for each decision, we consider not only local features about the pair, but also all of the past decisions made by our model on previously seen (entity,argument) pairs (see Table 4.1 for our full feature set). By doing so, we are able to take past decisions into context when filling up the template.

We begin by creating an initial training set of *(entity candidate, argument role)* pairs from our training documents, and learn a classifier (i.e. our policy π) to predict the correct argument roles for each candidate. We then expand our training set with additional examples, obtained by exploring the training data in the following fashion:

1. Randomly sample one of the documents from the training set
2. Follow the current learned policy π on this document until a randomly sampled step t
3. Generate a new training example using state s_t, where the reward for each possible action is determined by the final loss achieved when rolling out with the optimal policy π^* provided by an oracle. Add the new training example to the training set.
4. After generating k new examples, retrain the policy using the new training set, and jump back to step 1.

A key advantage of this approach is that we can consider the entire action history of a document while still remaining computationally tractable. By contrast, a linear chain conditional random field would be unable to draw upon the entire action history (due to the Markov property), and higher-order conditional random fields would become prohibitively expensive.

Local Features:
Entity name
Names of coreferent entities
Whether the entity is the first mention of a specific NER type
Paragraph number the entity occurs in
Gazetteers for common time expressions
Manually constructed context keyword lists for the target domain
Word embedding vectors
Dependent/governor information from dependency parsing
Global Features:
Previous classification decisions with the same entity on a different argument slot
Previous classification decisions on the same argument slot with other entities
Shared context words with entities from previous classification decisions

Table 4.1: Features used in our Learning to Search for Macro-Events model

4.2 Connections to Policy Gradient

In this section, we elucidate the connections of our proposed approach to policy gradient and the classic REINFORCE algorithm [78].

π_θ is parameterized by the parameters θ of the classifier, hence our goal is to find θ^* which maximizes the expected reward J_θ:

$$\theta^* = \text{argmax}_\theta \ J_\theta \tag{4.1}$$

$$= \text{argmax}_\theta \ \mathbb{E}_{\pi_\theta}\left[r(s_0, a_0, \cdots, s_T, a_T)\right] \tag{4.2}$$

where r is a binary reward function defined as

$$r(\cdot) = \begin{cases} 1 & \{a_t\}_{t=0}^T \text{ leads to the correct answer} \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Let R_T be output of $r(\cdot)$ at the last step. Taking the gradient w.r.t. the above objective function yields (using the log-derivative trick [78]):

$$\nabla_\theta J_\theta = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log p_\theta(s_0, a_0, \cdots, s_T, a_T) \cdot R_T\right] \tag{4.4}$$

where p_θ stands for the probability for rolling out experiences $s_0, a_0, \cdots, s_T, a_T$ under our current policy π_θ. Terms inside the derivative can be further factorized according to the Markov property:

$$\log p_\theta \cdot R_T = \log \prod_{t=0}^{T} p_\theta\left(s_t, a_t\right) R_T \tag{4.5}$$

$$= \sum_{t=0}^{T} \log p_\theta\left(s_t, a_t\right) y_t \tag{4.6}$$

where we have defined $y_0 = y_1 = \cdots = R_T$.

Equation (4.6) suggests maximizing J_θ is equivalent to minimizing the negative log-likelihood loss over the union of the original training data and newly generated pseudo-training examples $\{(s_t, a_t, y_t)\}$, where s_t, a_t and y_t denote the features of the training instance, the entity candidate and slot type (category) that we are currently looking at, and its associated label, respectively. As an example, when applying logistic loss for each category, the negative log-likelihood becomes

$$-\log p_\theta(s, a) \stackrel{def}{=} \log\left(1 + \exp(-\langle \theta_a, s \rangle)\right) \tag{4.7}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product; θ_a parameterizes the decision boundary for slot type a.

We would like to point out two differences between our implementation and the above analysis. First, the logistic loss is replaced with hinge loss due to its stronger empirical performance. Second, we relaxed the constraint that $y_0, y_1, ..., y_T$ have to be identical, allowing each action to have their individual reward based on its immediate feedback (namely whether the answer is correct for a particular slot). Unlike traditional supervised learning under the i.i.d. assumption, decisions made by our system are no longer independent as the state s dynamically evolves during the Markov process, which is crucial for capturing the structure both among and within different slot types.

4.3 Multilingual Macro-Event Extraction (Proposed Work)

We propose to expand this algorithm to multilingual capabilities, to demonstrate its use in languages beyond just English. In this extension, we will borrow from our previous work in multilingual ACE-style event extraction [33, 34, 35] and modify our macro-event extraction algorithm to multilingual capabilities. For data, we plan to extend the *election* macro-event data to an additional language. As Wikipedia already contains election articles (and corresponding infoboxes) written in non-English languages, we expect that this should provide a sufficient amount of data for multilingual experiments.

Chapter 5

Neural Machine Reading Comprehension

A disadvantage of training classifiers to perform information extraction is that new annotated data is required in order to deploy the algorithm to any new domain. To avoid this cost, we propose to leverage existing, large-scale machine reading comprehension corpora to train a general-purpose question answering system, and apply the resulting model directly to the task of macro-event extraction.

For the purposes of this work, we utilize the Gated-Attention (GA) reader[20]. The GA reader is a deep learning model that uses a multi-hop architecture combined with an attention mechanism, achieving state of the art results on multiple benchmark machine reading comprehension datasets. The multi-hop architecture simulates a human reading the document over several passes, each time refining the current understanding of the text. The attention mechanism serves to keep the reader focused on a given question while reading the text, allowing the model to assign greater importance to sections of the text that are of higher relevance.

5.1 Model Details

Figure 5.1 provides an illustration of the GA reader. The document and query are read over K layers, with each layer k taking the previous document embeddings from layer $k-1$ as input. Layer-specific document and query embeddings are each independently transformed using bi-directional Gated Recurrent Units (GRU) [16], and then combined using a Gated-Attention module. The Gated-Attention module is applied to each word embedding d_i in the document, in the following manner:

$$\begin{aligned} \alpha_i &= \text{softmax}(Q^T d_i) \\ \tilde{q}_i &= Q\alpha_i \\ x_i &= d_i \odot \tilde{q}_i \end{aligned} \tag{5.1}$$

where Q is the query embedding representation and $\odot$ is the Hadamard product.

At the final layer, the document and query representations are combined using an inner-product, and then run through a softmax layer to obtain a probability distribution over the tokens in the document. Probabilities are aggregated and renormalized for tokens that appear multiple

times in the document, and the final answer is obtained by selecting the candidate with maximum probability:

$$\begin{aligned} \Pr(c|d,q) &\propto \sum_{i \in \mathbb{I}(c,d)} s_i \\ a^* &= \arg\max_{c \in C} \Pr(c|d,q) \end{aligned} \tag{5.2}$$

where C is the set of candidate answers, s is the softmax probability vector, and $\mathbb{I}(c,d)$ designates the indexes of document d that correspond to candidate c.

5.2 Training Data

A variety of large-scale corpora for machine reading comprehension have been developed in recent years, including the CNN/DailyMail [30], SQuAD [67], and WDW datasets [63]. Each of these datasets contain (document, query, candidates, answer) tuples that can be used for training and evaluating machine reading comprehension techniques. Notably, the scale of each of these datasets is on the order of hundreds of thousands of examples. Any of these datasets can be used for training the GA Reader. For our work, we train on the WDW dataset, as its content focuses primarily on people and their actions, which we believe to well match the needs of event extraction.

5.3 Macro-Event Extraction

In order to run the GA Reader on a document, we need a natural language question and a set of candidate answers. The candidate answers are easily obtainable via entity recognition, leaving only the task of obtaining natural language questions. As the GA Reader is trained only on an off-domain corpus and does not see any annotated macro-event data, the model itself has no knowledge of either the macro-event structure or the roles that it needs to fill.

To overcome this, we handcraft natural language questions to pose to the GA Reader for each of the slots. Notably, as each macro-event type has a pre-defined structure, there are only a finite number of macro-event roles that the reader needs to account for, and therefore, only a finite number of natural language questions that need to be generated. For example, the *election* macro-event type only requires generating 6 different questions, and the *attack* macro-event only requires generating 5 questions. As a result, even though each argument role requires some human effort to create a question, it is in practice many orders of magnitude faster to accomplish this than to create on-domain training data through annotation. Example macro-event questions can be seen in Table 5.1.

5.4 Model Extensions (Proposed Work)

The major advantage of the current GA reader is that it can be applied to new domains easily, even when little or no on-domain training data exists. The downside however, is that it does not

GA Reader Questions		
Attack	Dead	@placeholder was killed or died in the attack or shooting
	Injured	@placeholder was wounded or injured in the attack or shooting
	Time	the attack or shooting occurred at @placeholder
Election	Winner	@placeholder won the election
	Loser	@placeholder lost the election
	Nominee	@placeholder was nominated to office

Table 5.1: Sample handcrafted questions passed to the GA Reader for macro-event extraction

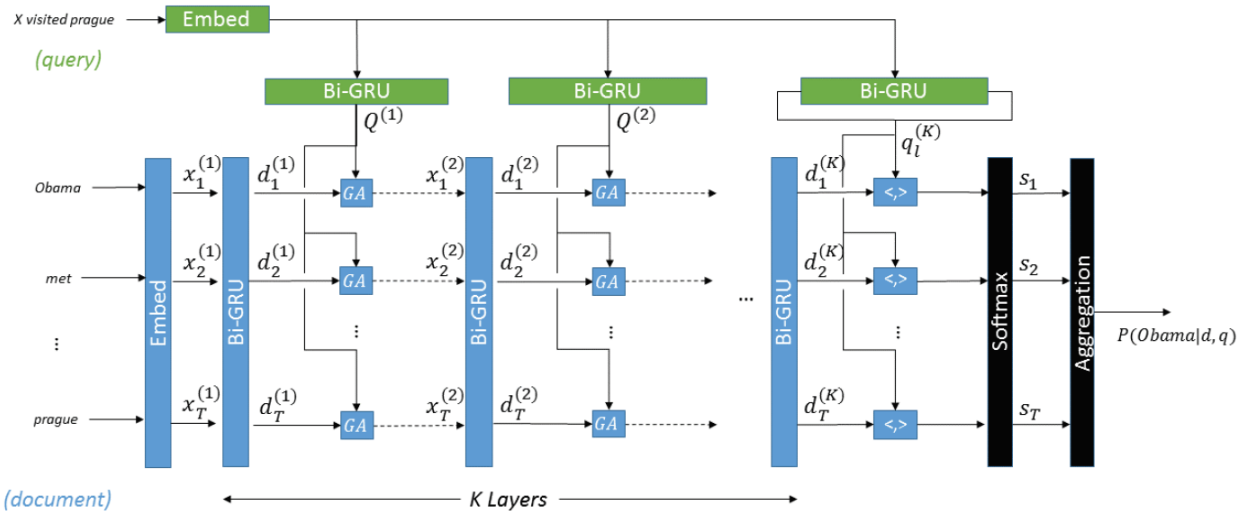


Figure 5.1: Architecture for the GA Reader. Each "GA" box represents a Gated-Attention module.

currently have any way of tuning its model parameters for domains where training data is readily available. As a part of our proposed work, we plan to explore methods of incorporating annotated macro-event training data into the GA Reader, thereby tuning our existing general-purpose GA reader model into a domain-specific GA reader.

Chapter 6

Preliminary Experiments

In this section, we will describe our preliminary experimental results for macro-event extraction. For the purposes of this study, we focus on two macro-event domains: *attacks* and *elections* (See Figures 3.1 and 3.2 for their hierarchical structures). We use the macro-event datasets described in Chapter 3

6.1 Baselines

We evaluate performance on our dataset using a variety of methods:

- First mention baseline – for each argument role, assigns the first named entity with valid entity type (e.g. a PERSON entity cannot fill the *Time* argument role)
- Most frequent baseline – for each argument role, assigns the most frequently mentioned named entity with valid entity type
- Aggregated Sentence-Level Event Extraction – runs ACE-style event extraction using joint inference of event triggers and arguments [47, 48, 49]. We directly use the implementation provided via the RPI Joint Information Extraction System[1]. For the *attack* dataset, we combine argument results from the *Conflict.Attack*, *Life.Injure*, and *Life.Die* event types into a single event template, while for the *elections* dataset, we combine argument results from the *Personnel.Elect* and *Personnel.Nominate* event types. To match our annotation scheme, we only consider arguments that are named entities.
- GA Reader – our proposed neural machine reading comprehension model for macro-events described in Chapter 5
- Independent Classifiers – a linear SVM using the local features described in Chapter 4
- Learning to Search for Macro-Events – our proposed model described in Chapter 4

[1] `http://nlp.cs.rpi.edu/software/`

	Attack		Elections	
	Micro-F1	Macro-F1	Micro-F1	Macro-F1
First Mention Baseline	31.9	29.3	47.1	46.0
Most Frequent Baseline	24.6	23.0	44.3	41.4
Aggregated Sentence-Level	30.1	21.4	23.8	19.8
GA Reader	42.0	33.1*	51.1	46.8
Independent Classifiers	40.9	29.9	74.6	71.6
Learning to Search for Macro-Events	**49.1**	**36.5**	**77.9**	**75.2**

Table 6.1: Results for Macro-Event Extraction. Bold entries represent the best performance in each column. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best model.

6.2 Experimental Setup

The *attack* dataset is split into three subsections by year of publication date. We use one year's data each for training, validation, and testing. For the *elections* dataset, we randomly partitioned the data into training, validation, and testing sets.

For both datasets, we process each document using Stanford CoreNLP [59] to obtain POS tags, named entities, entity coreference, and dependency parsing for our features.

The first mention and most frequent baselines do not require any training data. The aggregated sentence-level event extraction approach is trained using the ACE 2005 data. The GA Reader is trained using the WDW data. Lastly, the independent classifiers and Learning to Search methods are trained using the training set from our annotated data, both of which using LIBSVM[2] [5].

We evaluate all models using micro and macro-averaged F1 (the harmonic average of precision and recall).

6.3 Results

Results on both datasets can be seen in Table 6.1. Let us begin with our analysis of the results on the *attack* data. Among the two heuristic baselines, first mention outperforms most frequent mention. We suspect this is due to how news stories are written. In many cases, the first paragraph provides a high-level description of the main event (who, what, where, when), before going into more specific details. Note however, that while this is surprisingly effective for such a simple method, it remains insufficient to solve the problem.

We find that aggregation of sentence-level ACE event output performs quite poorly compared to even simple heuristics. Selecting the most frequent mention outperforms the sentence-level method on macro-averaged F1, while selecting the first mention for each role outperforms sentence-level aggregation on both macro and micro-averaged F1. This low performance indicates that sentence-level event extraction is currently insufficient for the macro-event extraction

[2]In principle one could use any binary classifier (e.g. logistic regression, perceptron) for both the independent classifiers and Learning to Search methods. Empirically, we found SVM gave the best performance.

task, and still has a long ways to go before it can be used for structured summarization.

Using independent classifiers provides a slight boost over the first mention baseline on the macro-averaged metric, but achieves an improvement of 28.2% on micro-averaged F1. The GA Reader provides an additional boost in performance over independent classifiers, with a 10.7% improvement on macro-averaged F1 and a 2.7% improvement on micro-averaged F1. Finally, our proposed Learning to Search method provides a further boost in performance, achieving an additional 10.3% improvement on macro-averaged F1 and 16.9% improvement on micro-averaged F1 over the neural method. Our top performing method of Learning to Search shows statistically significant improvement over independent classifiers on both micro-average and macro-average F1, and has statistically significant improvement over the GA Reader on micro-average F1.

Let us now consider the *elections* dataset. Aggregated sentence-level is the lowest performing method, and is even worse on *elections* than on *attacks*. We suspect that this is due to the distribution of training data in ACE2005, which contains more attack events than election events. The most frequent baseline performs almost twice as well as the sentence-level method, and the first mention baseline achieves a further improvement. Notably, both the most frequent and first mention baselines perform much strong on *elections* than on *attacks*, which suggests that this macro-event type is overall an easier domain.

The GA Reader is the next best method, with a slight improvement on macro-averaged F1 and an improvement of 8.5% on micro-averaged F1. The final two methods show a sizable boost in performance over all other methods, bring both macro-averaged and micro-averaged scores well into the 70s. Learning to Search ultimately has the strongest performance on this data, and shows statistically significant improvement over all other methods on both micro-averaged and macro-averaged F1.

While the GA Reader works quite well on the *attack* data, it lags behind independent classifiers and Learning to Search on *elections*. We believe this to be an effect of the much larger training data offered to these methods on the *elections data*, which is an order of magnitude larger than that of the *attack* data.

To more clearly see the effects of available training data on the three top methods, let us consider their performance on the *attack* data as the amount of available training data varies. Figures 6.1 and 6.2 show the micro-averaged and macro-averaged F1 results on the *attack* dataset across various percentages of training data. While the Learning to Search method clearly performs strongest when using the full training data, the performance quickly drops below that of the GA Reader when less training data is available. With less than 80% of the training data, the GA Reader performs stronger than Learning to Search on micro-averaged F1, and with less than 90% of the training data, the GA Reader performs stronger on macro-averaged F1. This clearly showcases the strength of the GA Reader on domains with little to no training data. Even more advantageous is that this performance on data-scarce domains comes at very little cost – once the GA Reader has been trained on its source corpus (e.g. WDW), there is very little additional effort required to port the model to test on a new domain. In contrast, both independent classifiers and Learning to Search require collection and annotation of new training data in order to deploy these methods on new domains.

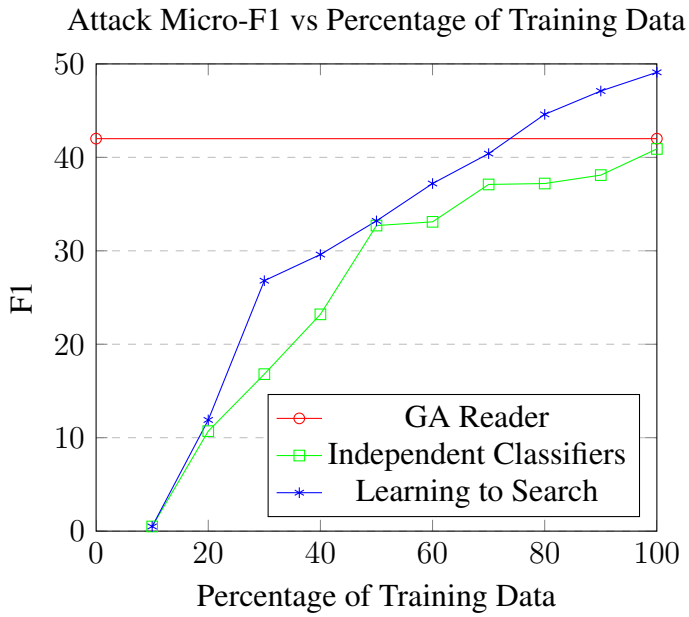


Figure 6.1: Micro-F1 results on the attack domain as the percentage of training data used varies.

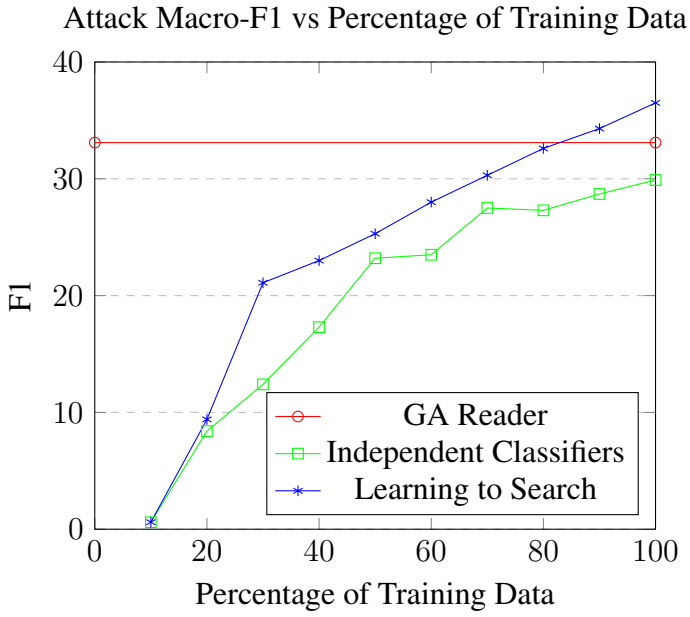


Figure 6.2: Macro-F1 results on the attack domain as the percentage of training data used varies.

Chapter 7

Timeline

The timeline for completion of this thesis is as follows:

- **July 2017 - September 2017**: Construction of macro-event annotated datasets for remaining domains (including at least one non-English dataset). As some of this will be completed by annotators, we expect to conduct this in parallel with other work.
- **July 2017 - September 2017**: Extension of our machine reading comprehension model to utilize available macro-event labeled data for domain adaptation. [Optional stretch target goal]
- **October 2017**: Adaptation of Learning to Search model to handle non-English texts
- **November 2017**: Completion of macro-event extraction experiments on all domains
- **October 2017 - December 2017**: Thesis writing, concurrent with experiment completion
- **January 2018**: Thesis defense

Chapter 8

Conclusion

In this thesis, we have proposed the macro-event framework, a new paradigm for event extraction designed to support structured summarization of news articles. Our goal is to push toward event extraction models capable of creating document-level summaries that can be useful to real-world users.

We have shown that existing frameworks for event extraction do not yet realize this goal, and introduced two methods for addressing this problem: 1.) a structured model that jointly learns the argument slots of a macro-event, and 2.) a neural machine reading comprehension model that requires zero available training data in the target domain. Our preliminary experimental results show that jointly learning the arguments for this task can achieve significantly improved performance over independent classifiers and other baseline methods, while applying deep neural networks for machine reading comprehension can achieve high performance for domains with little available training data.

In our proposed work, we hope to expand macro-events in several directions. First, we plan to extend macro-event extraction to several additional domains, to demonstrate that our approaches are generalizable. Second, we will demonstrate that our approaches can be applied to multilingual settings by deploying our extraction techniques to non-English texts for one of our domains. Finally, while our neural model achieved high performance on domains with little training data, the current model does not have any ability to take advantage of cases where the domain does have substantial training data available. We hope to explore ways of incorporating such data into the training of machine reading neural networks, in order to tune these general-knowledge models toward particular domains of interest.

Bibliography

[1] Jun Araki, Zhengzhong Liu, Eduard Hovy, and Teruko Mitamura. Detecting subevent structure for event coreference resolution. In *LREC*, 2014. 1.1

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 2003. 1.1

[3] Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998. 1.1, 2.3

[4] Asli Celikyilmaz and Dilek Hakkani-Tür. Discovery of topically coherent sentences for extractive summarization. In *ACL*, 2011. 1.1, 2.3

[5] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011. 6.2

[6] Eugene Charniak. Toward a model of childrens story comprehension. Technical report, MIT Artificial Intelligence Laboratory, 1972. 2.2

[7] Eugene Charniak, Yasemin Altun, Rodrigo de Salvo Braz, Benjamin Garrett, Margaret Kosmala, Tomer Moscovich, Lixin Pang, Changhee Pyo, Ye Sun, Wei Wy, Zhongfa Yang, Shawn Zeller, and Lisa Zorn. Reading comprehension programs in a statistical-language-processing class. In *ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding systems*, 2000. 2.2

[8] Chen Chen and Vincent Ng. Joint modeling for chinese event extraction with rich linguistic features. In *COLING*, 2012. 2.1

[9] Chen Chen and Vincent Ng. Sinocoreferencer: An end-to-end chinese event coreference resolver. In *LREC*, 2014. 2.1

[10] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*, 2016. 2.2

[11] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *ACL*, 2015. doi: 10.3115/v1/P15-1017. 1.1, 2.1

[12] Zheng Chen and Heng Ji. Language specific issue and feature exploration in chinese event extraction. In *HLT-NAACL*, 2009. 2.1

[13] Zheng Chen and Heng Ji. Can one language bootstrap the other: A case study on event extraction. In *NAACL HLT Workshop on Semi-supervised Learning for Natural Language Processing*, 2009. 2.1

[14] Hai Leong Chieu, Hwee Tou Ng, and Yoong Keok Lee. Closing the gap: Learning-based information extraction rivaling knowledge-engineering methods. In *ACL*, 2003. doi: 10.3115/1075096.1075124. 1.1, 2.1

[15] Nancy Chinchor, Lynette Hirschman, and David D. Lewis. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Computational Linguistics*, 1993. 1.1

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *ACL*, 2015. 5.1

[17] John M. Conroy and Dianne P. O'leary. Text summarization via hidden markov models. In *SIGIR*, 2001. 1.1, 2.3

[18] Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. Attention-over-attention neural networks for reading comprehension. *CoRR*, abs/1607.04423, 2016. URL `http://arxiv.org/abs/1607.04423`. 2.2

[19] Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. In *Machine Learning Journal*, 2009. 1.1

[20] Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016. 2.2, 5

[21] Günes Erkan and Dragomir R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of Artificial Intelligence Research*, 2004. 1.1, 2.3

[22] Oren Etzioni, Michele Banko, and Michael J Cafarella. Machine reading. In *AAAI*, 2006. 2.2

[23] Elena Filatova and Vasileios Hatzivassiloglou. Event-based extractive summarization. In *Proceedings of ACL Workshop on Summarization*, 2004. 2.3

[24] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: A graph-based approach to abstractive summarization of highly redundant opinions. In *COLING*, 2010. 1.1, 2.3

[25] Pierre-Etienne Genest and Guy Lapalme. Framework for abstractive summarization using text-to-text generation. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, 2011. 1.1, 2.3

[26] Pierre-Etienne Genest and Guy Lapalme. Fully abstractive approach to guided summarization. In *ACL*, 2012. 1.1, 2.3

[27] Ralph Grishman. The nyu system for muc-6 or where's the syntax? In *Proceedings of the 6th Conference on Message Understanding*, 1995. 1.1, 2.1

[28] Ralph Grishman and Beth Sundheim. Message understanding conference-6: A brief history. In *COLING*, 1996. doi: 10.3115/992628.992709. 1.1

[29] Ralph Grishman, David Westbrook, and Adam Meyers. Nyus english ace 2005 system description. In *Proc. ACE 2005 Evaluation Workshop*, 2005. 1.1, 2.1

[30] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay,

Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, 2015. 2.2, 5.2

[31] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*, 2016. 2.2

[32] Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. Deep read: A reading comprehension system. In *ACL*, 1999. 2.2

[33] Andrew Hsi, Jaime Carbonell, and Yiming Yang. Modeling event extraction via multilingual data sources. In *Text Analysis Conference (TAC 2015)*, 2015. 2.1, 4.3

[34] Andrew Hsi, Jaime Carbonell, and Yiming Yang. Cmu_cs_event tac-kbp2016 event argument extraction system. In *Text Analysis Conference (TAC 2016)*, 2016. 2.1, 4.3

[35] Andrew Hsi, Yiming Yang, Jaime Carbonell, and Ruochen Xu. Leveraging multilingual training for limited resource event extraction. In *COLING*, 2016. 2.1, 4.3

[36] Paul S. Jacobs, George Krupka, Lisa Rau, Michael L. Mauldin, Teruko Mitamura, Tsuyoshi Kitani, Ira Sider, and Lois Childs. The tipster/shogun project. In *Proceedings of the TIPSTER Text Program, Phase One*, 1993. 1.1, 2.1, 2.1

[37] Paul S Jacobs, George Krupka, Lisa Rau, Michael L Mauldin, Teruko Mitamura, Tsuyoshi Kitani, Ira Sider, and Lois Childs. Ge-cmu: Description of the shogun system used for muc-5. In *Proceedings of the 5th conference on Message understanding*, 1993. 1.1, 2.1, 2.1

[38] Heng Ji and Ralph Grishman. Refining event extraction through cross-document inference. In *ACL*, 2008. 1.1, 2.1

[39] Heng Ji, Benoit Favre, Wen-Pin Lin, Dan Gillick, Dilek Hakkani-Tur, and Ralph Grishman. Open-domain multi-document summarization via information extraction: Challenges and prospects. In *Multi-source, Multilingual Information Extraction and Summarization*. 2013. 2.3

[40] Hongyan Jing and Kathleen R. McKeown. Cut and paste based text summarization. In *NAACL*, 2000. 1.1, 2.3

[41] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *ACL*, 2016. 2.2

[42] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996. 4.1

[43] Dustin Lange, Christoph Böhm, and Felix Naumann. *Extracting structured information from Wikipedia articles to populate infoboxes*. 2010. ISBN 978-3-86956-081-6. 3.3.2

[44] Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. Joint entity and event coreference resolution across documents. In *EMNLP-CONLL*, 2012. 1.1

[45] Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 2013. 1.1

[46] Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. Employing compositional

semantics and discourse consistency in chinese event extraction. In *EMNLP*, 2012. 2.1

[47] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL*, 2014. doi: 10.3115/v1/P14-1038. 6.1

[48] Qi Li, Heng Ji, and Liang Huang. Joint event extraction via structured prediction with global features. In *ACL*, 2013. 1.1, 2.1, 6.1

[49] Qi Li, Heng Ji, Yu Hong, and Sujian Li. Constructing information networks using one single model. In *EMNLP*, 2014. doi: 10.3115/v1/D14-1198. 1.1, 2.1, 6.1

[50] Shasha Liao and Ralph Grishman. Filtered ranking for bootstrapping in event extraction. In *COLING*, 2010. 1.1, 2.1

[51] Shasha Liao and Ralph Grishman. Acquiring topic features to improve event extraction: in pre-selected and balanced collections. In *RANLP*, 2011. 1.1, 2.1

[52] Hui Lin and Jeff Bilmes. A class of submodular functions for document summarization. In *ACL*, 2011. 1.1, 2.3

[53] Marina Litvak, Mark Last, and Menahem Friedman. A new approach to improving multilingual summarization using a genetic algorithm. In *ACL*, 2010. 1.1, 2.3

[54] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. Toward abstractive summarization using semantic representations. In *NAACL*, 2015. 1.1, 2.3

[55] Jingen Liu, Qian Yu, Omar Javed, Saad Ali, Amir Tamrakar, Ajay Divakaran, Hui Cheng, and Harpreet Sawhney. Video event recognition using concept attributes. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 2013. 2.1

[56] Zhengzhong Liu, Jun Araki, Eduard Hovy, and Teruko Mitamura. Supervised within-document event coreference using information propagation. In *LREC*, 2014. 1.1

[57] Zhigang Ma, Yi Yang, Yang Cai, Nicu Sebe, and Alexander G. Hauptmann. Knowledge adaptation for ad hoc multimedia event detection with few exemplars. In *Proceedings of the 20th ACM International Conference on Multimedia*, 2012. 2.1

[58] Zhigang Ma, Yi Yang, Zhongwen Xu, Shuicheng Yan, Nicu Sebe, and Alexander G. Hauptmann. Complex event detection via multi-source video attributes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013. 2.1

[59] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, 2014. doi: 10.3115/v1/P14-5010. 6.2

[60] Michele Merler, Bert Huang, Lexing Xie, Gang Hua, and Apostol Natsev. Semantic model vectors for complex video event recognition. *IEEE Transactions on Multimedia*, 2012. 2.1

[61] Rada Mihalcea. Language independent extractive summarization. In *Proceedings of the ACL 2005 on Interactive Poster and Demonstration Sessions*, 2005. 1.1, 2.3

[62] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *ACL*, 2015. doi: 10.3115/v1/P15-2060. 1.1, 2.1

[63] Takeshi Onishi, Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Who did what: A large-scale person-centered cloze dataset. In *EMNLP*, 2016. doi: 10.18653/v1/

D16-1241. 2.2, 5.2

[64] Siddharth Patwardhan and Ellen Riloff. A unified model of phrasal and sentential evidence for information extraction. In *EMNLP*, 2009. doi: 10.3115/1699510.1699530. 1.1, 2.1

[65] Dragomir R. Radev and Kathleen R. McKeown. Generating natural language summaries from multiple on-line sources. *Computational Linguistics*, 1998. 1.1, 2.3

[66] Dragomir R. Radev, Hongyan Jing, and Malgorzata Budzikowska. Centroid-based summarization of multiple documents: Sentence extraction, utility-based evaluation, and user studies. In *Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization*, 2000. 1.1, 2.3

[67] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016. 2.2, 5.2

[68] Vignesh Ramanathan, Percy Liang, and Li Fei-Fei. Video event understanding using natural language descriptions. In *The IEEE International Conference on Computer Vision (ICCV)*, 2013. 2.1

[69] Amjad Rehman and Tanzila Saba. Features extraction for soccer video semantic analysis: current achievements and remaining issues. *Artificial Intelligence Review*, 2014. 2.1

[70] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, 2013. 2.2

[71] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *AAAI*, 1996. 1.1, 2.1

[72] Ellen Riloff and Michael Thelen. A rule-based question answering system for reading comprehension tests. In *ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding systems*, 2000. 2.2

[73] Stéphane Ross and J. Andrew Bagnell. Reinforcement and imitation learning via interactive no-regret learning. 2014. arXiv:1406.5979. 4.1

[74] Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *EMNLP*, 2015. 1.1, 2.3

[75] Horacio Saggion and Guy Lapalme. Generating indicative-informative summaries with sumum. *Computational Linguistics*, 2002. 1.1, 2.3

[76] Mark Stevenson and Mark A Greenwood. A semantic approach to ie pattern induction. In *ACL*, 2005. doi: 10.3115/1219840.1219887. 1.1, 2.1

[77] Chen Sun and Ram Nevatia. Large-scale web video event classification by use of fisher vectors. In *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, 2013. 2.1

[78] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 4.2, 4.2

[79] Kam-Fai Wong, Mingli Wu, and Wenjie Li. Extractive summarization using supervised and semi-supervised learning. In *COLING*, 2008. 1.1, 2.3

[80] Zhongwen Xu, Yi Yang, and Alex G. Hauptmann. A discriminative cnn video representation for event detection. In *The IEEE Conference on Computer Vision and Pattern Recog-*

nition (CVPR), June 2015. 2.1

[81] Yan Yan, Yi Yang, Deyu Meng, Gaowen Liu, Wei Tong, Alexander G. Hauptmann, and Nicu Sebe. Event oriented dictionary learning for complex event detection. *IEEE Transactions on Image Processing*, 2015. 2.1

[82] Bishan Yang and Tom Mitchell. Joint extraction of events and entities within a document context. In *NAACL*, 2016. doi: 10.18653/v1/N16-1033. 1.1

[83] Roman Yangarber, Ralph Grishman, Pasi Tapanainen, and Silja Huttunen. Automatic acquisition of domain knowledge for information extraction. In *COLING*, 2000. doi: 10.3115/992730.992782. 1.1, 2.1