Event Extraction for Document-Level Structured Summarization

Andrew Hsi

CMU-LTI-18-002

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

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
In Language and Information Technologies

For Yuxin, whose unwavering support made this thesis possible.

Abstract

Event extraction has been well studied for more than two decades, through both the lens of document-level and sentence-level event extraction. 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 on document-level event extraction methods have focused on highly specific domains, often with great reliance on handcrafted rules. Such approaches do not generalize well to new domains. In contrast, sentence-level event extraction methods have applied to a much wider variety of domains, but generate output at such fine-grained details that they cannot offer good document-level summaries of events.

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 three novel methods to address the macro-event extraction task. 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 multi-layer neural network that is trained directly on macro-event annotated data. Finally, we propose a deep learning method that treats the problem as machine comprehension, which does not require training with any on-domain macro-event labeled data. Our experimental results on a variety of domains show that such algorithms can achieve stronger performance on this task compared to existing baseline approaches. On average across all datasets, neural networks can achieve a 1.76% and 3.96% improvement on micro-averaged and macro-averaged F1 respectively over baseline approaches, while Learning to Search achieves a 3.87% and 5.10% improvement over baseline approaches on the same metrics. Furthermore, under scenarios of limited training data, we find that machine comprehension models can offer very strong performance compared to directly supervised algorithms, while requiring very little human effort to adapt to new domains.

Acknowledgments

There are numerous people whose help and support were essential to the completion of this thesis. First and foremost are Yiming Yang and Jaime Carbonell, who have been my advisors since my first year at CMU. Their mentoring has been a truly wonderful experience, and I could not have gotten this far without them. I have learned so much from Yiming and Jaime over the course of my PhD and will always cherish their advice and insights. In addition to my advisors, I would also like to thank the rest of my thesis committee, Alex Hauptmann and Michael Mauldin. Their suggestions and feedback on my work have been invaluable, and I really appreciate all of their assistance. A special thank you also to Dan Roth, who first introduced me to machine learning as an undergrad at UIUC and later advised my undergraduate thesis work.

There are also many other people who I would like to thank for their support and contributions. Thank you to all of my collaborators, research group members, and friends in Pittsburgh, including Eduard Hovy, Teruko Mitamura, Hans Chalupsky, Edwin Morris, Siddharth Gopal, Hanxiao Liu, Wanli Ma, Guoqing Zheng, Ruochen Xu, Yuexin Wu, Wei-Cheng Chang, Jingzhou Liu, Guokun Lai, Naoki Otani, Bohan Li, Weiran Xu, Jun Araki, Zhengzhong Liu, Xuezhe Ma, Evangelia Spiliopoulou, Daegun Won, Petar Stojanov, Kevin Pitstick, Aubrey Henderson, Ben Bradshaw, Justin Chiu, Max Burstyn, Yubin Kim, Greg Hanneman, Alan Vee, Keith Bare, Kartik Goyal, Austin Matthews, and Barbara Davis.

A big thank you to my two best friends, Jun Cheng and Peter Chen, for helping me remember to take it easy once in a while and relax after a hard day's work. Game night is always a blast with you two!

I am eternally grateful to the support and love from my parents and grandparents, who always pushed me to do my best and have supported me all the way through my PhD.

Finally, a very special thank you to my beloved wife Yuxin Jiang, whose love and support always gets me through even the toughest of times. I am truly blessed to have you in my life.

Contents

1 Introduction **1**
- 1.1 Thesis Statement . . . 1
- 1.2 Thesis Contributions . . . 3
- 1.3 Thesis Outline . . . 4

2 Related Work **7**
- 2.1 Event Extraction . . . 7
  - 2.1.1 Document-Level Extraction . . . 7
  - 2.1.2 Sentence-Level Extraction . . . 12
  - 2.1.3 Event Extraction in Non-English Languages . . . 18
  - 2.1.4 Event Extraction in Video . . . 18
- 2.2 Machine Reading Comprehension . . . 19
- 2.3 Summarization . . . 20
  - 2.3.1 Extractive Summarization . . . 20
  - 2.3.2 Abstractive Summarization . . . 21

3 Macro-Events **23**
- 3.1 Definition . . . 23
- 3.2 Event Ontology . . . 27
  - 3.2.1 Attacks . . . 27
  - 3.2.2 Elections . . . 27
  - 3.2.3 Sporting Events . . . 28
  - 3.2.4 Criminal Trials . . . 28
- 3.3 Annotation . . . 29
  - 3.3.1 Leveraging Online Resources for Annotation . . . 29
  - 3.3.2 Internal Annotation . . . 31

4 Learning to Search **33**
- 4.1 Our Proposed Model . . . 33
- 4.2 Connections to Policy Gradient . . . 35

5 Deep Learning for Macro-Event Extraction **37**
- 5.1 Comparisons to Existing Work . . . 37
- 5.2 Model Details . . . 38

6 Neural Machine Reading Comprehension **43**
6.1 Model Details . . . 43
6.2 Training Data . . . 45
6.2.1 CNN/DailyMail . . . 45
6.2.2 CBT . . . 45
6.2.3 SQuAD . . . 45
6.2.4 WDW . . . 46
6.2.5 Comparison of Datasets . . . 47
6.3 Macro-Event Extraction . . . 47

7 Experimental Results **49**
7.1 Baselines . . . 49
7.2 Experimental Setup . . . 50
7.3 Results . . . 51
7.3.1 Attacks . . . 51
7.3.2 Sporting-Events . . . 51
7.3.3 Criminal-Trials . . . 52
7.3.4 Elections . . . 53
7.4 Discussion of Results . . . 55
7.5 Error Analysis . . . 55
7.6 Experiments with Limited Training Data . . . 57

8 Conclusion **63**

Bibliography **67**

List of Figures

2.1 Example input text to FRUMP 8
2.2 Output results from FRUMP on Figure 2.1's text 8
2.3 Example MUC template from the Latin American Terrorism domain 9
2.4 Factor graph for within-event structures used by Yang and Mitchell [2016]. t_i represents the event type (e.g. Life.Injure, Conflict.Demonstration) of trigger candidate i. $a_1 ... a_m$ represent the entity candidates occurring within the same sentence as trigger candidate i. $r_{i,j}$ represents the argument role type (if any) existing between i and a_j (e.g. Place, Time, Attacker). 16
2.5 ConvNet model used by Nguyen and Grishman [2015]. The convolutional layer allows the model to construct feature maps from words and their contexts. The motivation for the use of ConvNets is to capture information reflecting higher-level semantics, as opposed to solely individual word-level features 17
2.6 ConvNet model used by Chen et al. [2015]. The overall intuition is the same as in the model of Nguyen and Grishman [2015], but is additionally applied to argument extraction, whereas the Nguyen and Grishman model is only used for trigger extraction. 18

3.1 Hierarchy defined by the *attack* macro-event type 24
3.2 Hierarchy defined by the *election* macro-event type 26
3.3 Hierarchy defined by the *transaction* macro-event type 27
3.4 Hierarchy defined by the *sporting-event* macro-event type 28
3.5 Hierarchy defined by the *criminal-trial* macro-event type 29
3.6 Example Wikipedia infobox and its corresponding document 30

5.1 Architecture for neural network-based macro-event extraction. Input features are extracted from the document and current *(entity,role)* pair. The input is then fed through multiple hidden layers with ReLU activation functions to add nonlinearity. At the output layer, a binary decision is made to either include or exclude the candidate pair from the macro-event template. 38
5.2 Visualization of the sigmoid function 39
5.3 Visualization of the hyperbolic tangent function 40
5.4 Visualization of a rectified linear unit (ReLU) 40

6.1 Architecture for the GA Reader. Embeddings for both the query and document words are obtained via look-up tables, and then passed through bi-directional GRUs to obtain a transformed representation. Each "GA" box represents a Gated-Attention module, which apply query-focused attention to the document representation. After repeating this process over multiple layers, a score is computed for each word in the document. These scores are converted to a probability distribution over the words, and probabilities for repeated words in the text are aggregated. The final resulting probability distribution is used to select answers for the query. 44
6.2 Process by which (document, question, answer) tuples are generated for CNN data. 46

7.1 Number of training examples for each category in the *elections* domain. 57
7.2 Performance by the top three methods on individual categories in the *elections* data. 58
7.3 Number of training examples for each category in the *attacks* domain. 59
7.4 Performance by the top three methods on individual categories in the *attacks* data. 60
7.5 Micro-F1 results on the attack domain as the percentage of training data used varies. 61
7.6 Macro-F1 results on the attack domain as the percentage of training data used varies. 62

List of Tables

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. 4

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). 5

2.1 Event types studied in the MUC program by year 9

2.2 Event types and subtypes considered in the ACE program 19

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. 23

3.2 An example filled macro-event template for the *attack* domain. 25

3.3 A more complicated *attack* macro-event. This document describes a shooter who killed multiple people in one night, and injured several others. The macro-event structure attempts to summarize the entire attack, rather than treating each individual incident as a separate event (as would be done in sentence-level extraction). 26

3.4 A gold-standard macro-event template extracted from the infobox seen in Figure 3.6. 30

3.5 Statistics for macro-event corpora 32

4.1 Features used in our Learning to Search for Macro-Events model 35

6.1 Information on each of the major datasets for machine reading comprehension. . 48

6.2 Sample handcrafted questions passed to the GA Reader for macro-event extraction 48

7.1 Training data requirements for macro-event extraction algorithms 50

7.2 Results from macro-event extraction on the *attacks* domain. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method. 51

7.3 Results from macro-event extraction on the *sporting-events* domain. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method. . . 52
7.4 Results from macro-event extraction on the *criminal-trials* domain. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method. . . 53
7.5 Results from macro-event extraction on the *elections* domain, English documents. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method. 54
7.6 Results from macro-event extraction on the *elections* domain, Spanish documents. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method. 54

Chapter 1

Introduction

1.1 Thesis Statement

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 one of two different paradigms: sentence-level event extraction and document-level extraction (for a detailed discussion, please refer to Chapter 2). Most recent work follows sentence-level extraction, and in particular following the standards set by the Automatic Content Extraction (ACE) program[1] [17, 43, 57, 72, 73, 75, 92, 128]. The sentence-level focus means identifying individual event mentions from potentially every sentence in the document, along with any entities which fulfill argument roles in these events. While the event types considered via sentence-level extraction are general enough to be of interest to common users, the sentence-level granularity is 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). A further complication is that any meaningful downstream processing of sentence-level output requires high-quality event coreference algorithms, as a document may contain multiple references to the same event. Consider the following sentences:

[1] `http://www.itl.nist.gov/iad/mig/tests/ace/`

1. John Smith was arrested yesterday for the murder of Jane Doe.
2. The Boston murder from last week had stumped police until new evidence had been found connecting Smith to the victim.

While these two sentences both describe the same attack, each sentence provides unique information about the event arguments. The former sentence gives the full name of the perpetrator and victim, but lacks information about where and when the attack occurred. The latter sentence does contain the missing information, but lacks the full name of the perpetrator and provides no information whatsoever on the identity of the victim. Recent work has attempted to solve the event coreference problem [3, 5, 67, 81], however such methods are imperfect, and can lead to compounding errors in an NLP pipeline.

In contrast to sentence-level extraction, document-level event templates focus on a single template per document, and thus are able to achieve a more desirable level of granularity for downstream consumption. Document-level event extraction dates back to FRUMP (Fast Reading Understanding and Memory Program) [30], which constructed event templates from text using handcrafted rules. This provided the right level of granularity for template-driven summaries of documents, but was limited by its reliance on knowledge-engineering. Improved event coverage could only be obtained via extensive human effort, which is not practical for real-world deployment. More recent document-level event extraction techniques follow the definitions provided by the Message Understanding Conferences[2] (MUC) [2, 4, 7, 20, 21, 41, 42, 52, 53, 54, 55, 56, 63, 69, 74, 95, 96, 97, 105, 106, 116, 117, 118, 119, 121, 129, 130]. However, work to date on document-level extraction under the MUC definitions suffers from several limitations. First, the set of event types studied under document-level extraction via MUC are highly specific to a small set of focused domains (e.g. terrorist events in Latin America, electronic circuit fabrication, and launches of rockets/missiles). This limits the generalizability of document-level extraction systems, as such topics do not provide wide enough coverage over common public interests. A second key limitation is the lack of sophisticated machine learning techniques for document-level extraction. Early document-level extraction methods under MUC relied strongly on handcrafted rules (similarly to FRUMP) or pattern matching techniques. More recent work has attempted to use machine learning, but to date such work has focused on simple machine learning techniques, and do not take advantage of more powerful machine learning algorithms.

In this thesis, we propose a new event extraction task for extracting document-level structured summaries, which we call *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 sentence-level event extraction would help the solutions for the macro-event task. However, it would be unrealistic to rely on near-perfect sentence-level event extraction, near-perfect sentence-level event coreference [3, 5, 67, 81], near-perfect entity coref-

[2] `http://www-nlpir.nist.gov/related_projects/muc/`

erence [67, 68], near-perfect aggregation of sentence-level events to a document-level template, and near-perfect information extraction across entities, event triggers, and event arguments (all of which are vulnerable to compounding errors in the processing pipeline). Existing sentence-level extraction algorithms for these subtasks frequently produce outputs containing incorrect extractions, off-topic extractions, contradictory extractions, redundancy, and empty events devoid of argument fillers (see Table 1.2 for examples of these problems). Thus it is better to focus directly on the macro-event task instead, without requiring 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 [29] that addresses all three of these problems, achieving high performance on the macro-event extraction task. We further introduce two novel approaches for neural document-level event extraction: 1.) a feedforward neural network trained directly on annotated data, and 2.) a neural machine reading comprehension model that can achieve high performance over baseline methods even when little to no training data exists for the target 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 (Latent Dirichlet allocation) [6] have latent factors that are difficult to interpret, and do not offer insight into the fillers of arguments. Both extractive [8, 9, 27, 35, 76, 77, 90, 91, 99, 125] and abstractive summarization [38, 39, 40, 59, 79, 88, 98, 110, 111] 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.

1.2 Thesis Contributions

The overall contributions of this thesis are as follows:

- 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.

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.

- Development of multiple deep learning models for macro-event extraction, including a model which can be trained entirely on off-domain data
- Collection and annotation of macro-event gold standard data for the *attack*, *election*, *sporting-event* and *criminal-trial* domains. Labels have been obtained using a combination of manual annotation and infobox extraction from Wikipedia articles. For the *elections* domain, we have additionally collected Spanish language data to demonstrate the effectiveness of our models on non-English texts
- Experimental results on the *attack*, *election*, *sporting-event* and *criminal-trial* domains with baseline methods and all three of our proposed algorithms.

1.3 Thesis Outline

The remainder of this thesis 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 structured model for filling up macro-events, and describe connections between this approach and policy gradients. In Chapters 5 and 6, we introduce our deep learning models, one directly trained on macro-event annotated data, and another based on recent advances in neural machine reading techniques with zero reliance on target domain annotated training data. Chapter 7 describes our experimental results on filling macro-events from text on five separate macro-event datasets. We provide concluding thoughts and ideas for future work 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

In this chapter, we will introduce relevant literature in event extraction, machine reading comprehension, and summarization.

2.1 Event Extraction

In this section, we will discuss relevant related work in the field of event extraction, including both document-level and sentence-level extraction methods.

2.1.1 Document-Level Extraction

FRUMP

The earliest event extraction system was FRUMP [30]. The goal of FRUMP was to skim input news articles and extract events describing the most important aspects of the text. FRUMP achieved this by using two components – a predictor and a substantiator – to collaboratively fill up an event template. The predictor would be in charge of predicting what event frames exist in the text, while the substantiator would find evidence to fill up frames suggested by the predictor. Based on the evidence received from the substantiator, the predictor could then narrow down the set of possible candidate event templates, and issue new requests to the substantiator to fill up remaining slots in the template.

Consider the example text in Figure 2.1. FRUMP begins by scanning the text for a word that triggers the creation of an event template. Upon finding the word "crashed", FRUMP starts to explore possible "crash"-related event templates. The predictor makes a request to the substantiator to identify an actor for the "PROPEL" action, which can be one of many different filler types (e.g. "HUMAN", "VEHICLE", "MILITARY UNIT", etc.). The substantiator looks in the text for a span that can fill this role, and returns "plane" as the filler. Based on the evidence returned from the substantiator, the predictor can now narrow down the set of possible event types to two possibilities – either a vehicle hitting a person or a vehicle hitting some physical object. Accordingly, the predictor requests the substantiator to find evidence for what the plane hit, which must be either of type "HUMAN" or of type "PHYSOBJ". The predictor and substantiator continue such

```
     COLORADO (UPI)-RESCUERS ON SNOWCATS
PUSHED THROUGH FIVE-FOOT SNOWDRIFTS TO
REACH THE WRECKAGE OF A TWINENGINE ROCKY
MOUNTAIN AIRWAYS PLANE THAT CRASHED IN
COLORADO'S RUGGED BUFFALO PASS. AUTHORITIES
REPORTED ONE PERSON HAD BEEN KILLED BUT
THAT 21 OTHERS ABOARD SURVIVED. AMBULANCES
WERE ORDERED TO REMOVE THEM FROM THE
WILDERNESS.
```

Figure 2.1: Example input text to FRUMP

efforts until the complete event template is extracted (see Figure 2.2 for the final result extracted from FRUMP on this text).

```
($vehicle-accident
   (&obj1 (*plane*))
   (&obj2 (*ground*))
   (loc (*colorado*)))

(|casualty
   (&deadgrp (*human* (quant (1)))
   (&hurtgrp nil)
   (&missinggrp nil))

(cd
   (actor (*plane*))
   (<=> (*propel*))
   (object (*ground*))
   (loc (*colorado*))
   (manner (*violent*)))

(cd
   (actor (*human* (quant (1))))
   (is (*health* (value (-10)))))
```

Figure 2.2: Output results from FRUMP on Figure 2.1's text

A key limitation of FRUMP is that it requires human-crafted knowledge in order to make decisions. While a system like FRUMP can in principle perform very well on a domain where much knowledge has been injected, it is impractical to port such a system to new domains. In contrast, our proposed methods are designed to require a minimal amount of human effort by comparison, and are much easier to apply to new domains of interest.

Pattern Matching

The DARPA MUC program (1987-1997) brought considerable interest to the topic of document-level event extraction. 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). An example MUC template from the Latin American Terrorism domain may be seen in Figure 2.3.

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

```
0.  MESSAGE: ID                 DEV-MUC3-1101 (UNISYS / PARAMAX)
1.  MESSAGE: TEMPLATE           1
2.  INCIDENT: DATE              11 Feb 90
3.  INCIDENT: LOCATION          EL SALVADOR: CHALATENANGO (DEPARTMENT): CORRAL DE LAS PIEDRAS (REFUGEE CAMP)
4.  INCIDENT: TYPE              ATTACK
5.  INCIDENT: STAGE OF EXECUTION    ACCOMPLISHED
6.  INCIDENT: INSTRUMENT ID     "ROCKETS" / "ROCKET"
                                "GRENADE-TYPE DEVICE"
                                "MACHINEGUN"
7.  INCIDENT: INSTRUMENT TYPE   ROCKET: "ROCKETS" / "ROCKET"
                                GRENADE: "GRENADE-TYPE DEVICE"
                                MACHINE GUN: "MACHINEGUN"
8.  PERP: INCIDENT CATEGORY     STATE-SPONSORED VIOLENCE
9.  PERP: INDIVIDUAL ID         -
10. PERP: ORGANIZATION ID       "THE SALVADORAN ARMED FORCES" / "SALVADORAN ARMED FORCES" / "ARMED FORCES"
11. PERP: ORGANIZATION CONFIDENCE   SUSPECTED OR ACCUSED: "THE SALVADORAN ARMED FORCES" / "SALVADORAN ARMED FORCES" / "ARMED FORCES"
12. PHYS TGT: ID                "A REFUGEE CAMP" / "REFUGEE CAMP" / "CAMP"
13. PHYS TGT: TYPE              OTHER: "A REFUGEE CAMP" / "REFUGEE CAMP" / "CAMP"
14. PHYS TGT: NUMBER            1: "A REFUGEE CAMP" / "REFUGEE CAMP" / "CAMP"
15. PHYS TGT: FOREIGN NATION    -
16. PHYS TGT: EFFECT OF INCIDENT    SOME DAMAGE: "A REFUGEE CAMP" / "REFUGEE CAMP" / "CAMP"
17. PHYS TGT: TOTAL NUMBER      -
18. HUM TGT: NAME               -
19. HUM TGT: DESCRIPTION        "PEOPLE"
                                "CIVILIANS"
20. HUM TGT: TYPE               CIVILIAN: "CIVILIANS"
21. HUM TGT: NUMBER             5: "PEOPLE"
                                14-: "CIVILIANS"
22. HUM TGT: FOREIGN NATION     -
23. HUM TGT: EFFECT OF INCIDENT     DEATH: "PEOPLE"
                                INJURY: "CIVILIANS"
24. HUM TGT: TOTAL NUMBER       19
```

Figure 2.3: Example MUC template from the Latin American Terrorism domain

Early methods developed during this time typically focused on handcrafted patterns [2, 69, 121]. The basic idea behind such approaches is to scan the text for particular expressions, which can directly map pieces of the document into argument role slots. Several example patterns included in the FASTUS system [2] for the Latin American terrorism domain may be seen below:

1. killing of ⟨HumanTarget⟩

2. ⟨GovtOfficial⟩ accused ⟨PerpOrg⟩
3. bomb was placed by ⟨Perp⟩ on ⟨PhysicalTarget⟩
4. ⟨Perp⟩ attacked ⟨HumanTarget⟩'s ⟨PhysicalTarget⟩ in ⟨Location⟩ ⟨Date⟩ with ⟨Device⟩
5. ⟨HumanTarget⟩ was injured
6. ⟨HumanTarget⟩'s body

where the parts contained inside the brackets reference particular argument slots in the event template type. If we consider the sentence "Guerillas attacked Merino's home in San Salvador 5 days ago with explosives", we can see that this matches the above pattern "⟨Perp⟩ attacked ⟨HumanTarget⟩'s ⟨PhysicalTarget⟩ in ⟨Location⟩ ⟨Date⟩ with ⟨Device⟩". By matching the text to this pattern, we can extract the following slots:

1. Perp: Guerillas
2. HumanTarget: Merino
3. PhysicalTarget: home
4. Location: San Salvador
5. Date: 5 days ago
6. Device: explosives

Note that such methods can extract multi-word phrases, such as "San Salvador" and "5 days ago", and are thus not limited to only individual words. A key limitation of such methods is that this requires handcrafted human effort, which makes it extremely difficult to adapt systems to new domains.

In response to this problem Riloff [105] designed a system called AutoSlog which automatically generates extraction patterns given a corpus of annotated data. In order to avoid low-quality patterns, a human was kept in the loop to manually check each discovered pattern, editing or removing those that were deemed to be poor or incorrect. This process required 5 hours, and resulted in a final set of 450 discovered patterns (from an original set of 1237 proposed patterns). In their experiments on the MUC-4 dataset, they found that this gave nearly as good of performance as a fully handcrafted system that required 1500 person hours to construct. A number of similar systems for automatically learning patterns were also proposed, including PALKA [63], CRYSTAL [116], and LIEP [54].

AutoSlog was later expanded to generate patterns without reliance on annotated text, given only two separate corpora of relevant and irrelevant texts to the target domain [106]. The system, called AutoSlog-TS, operates in a two-stage manner. First, it generates pattern candidates for every noun phrase in the relevant corpus. In the second step, these patterns are applied to both the relevant and irrelevant documents, and are ranked according to the following formula:

$$\Pr(\text{relevant text}|\text{text contains pattern i}) = \frac{rel - freq_i}{total - freq_i}$$

where $rel - freq_i$ is simply the number of times the pattern was activated in the relevant documents, and $total - freq_i$ is the number of times the pattern was activated across both corpora. The general idea is that some phrases will occur frequently regardless of domain (e.g. "was reported") whereas others are more domain specific (e.g. "was kidnapped").

A human was then used to review the top patterns according to this ranking; they stopped after analyzing the top 1970 patterns as few remaining patterns were of high quality. This resulted in a total of 210 extracted patterns. Evaluation on MUC-4 documents against AutoSlog showed improved performance on precision and F1, but lower recall.

Yangarber et al. [2000] proposed an alternative to AutoSlog-TS called EXDISCO. Given a small seed set of patterns, EXDISCO operates using the following procedure:

1. Apply the patterns to all documents in the collection. Separate documents into relevant and non-relevant based on whether any patterns were matched within the text.
2. Generate new candidate patterns from the relevant documents. Rank them by relevance scores (similarly to AutoSlog-TS).
3. Given the ranked list of generated patterns, add the top pattern to the seed set
4. Repeat from step 1

The key difference from AutoSlog-TS is that EXDISCO does not require having human judgements for relevant and non-relevant documents, as these are automatically categorized based on the seed set of patterns.

Other work has also considered the task of automatically generating patterns from unannotated text. Sudo et al. [2001, 2003] consider patterns based on chains and subtrees from dependency parsing output. Yangarber [2003] explore stopping conditions for automatic pattern generation algorithms, which over time begin to generate fewer and fewer high-quality patterns. Patwardhan and Riloff [2006] expand the AutoSlog-TS model to identify additional patterns from unannotated data from the Web. Later work by Patwardhan and Riloff [2007] combines automatic generation of patterns with a self-trained relevance classifier for sentences[1]. Patterns are thresholded into primary and secondary patterns – at test time, primary patterns are used to extract information from all sentences, while secondary patterns are extracted only from sentences deemed to be relevant by the classifier.

Classification

While most of the information extraction community during the 1990s was focused on knowledge-based approaches, there was some effort to move toward machine learning models, which had the advantage of being much more adaptable to new domains. However, such models were not able to outperform the results obtained by knowledge-based approaches [4, 69].

The first classification-based method shown to achieve competitive performance with pattern-based methods on the MUC data was that of Chieu et al. [2003], via a system called ALICE (Automated Learning-based Information Content Extraction). ALICE begins by first preprocessing input documents with NLP software for sentence segmentation, tokenization, part-of-speech tagging, named entity recognition, parsing, and entity coreference. Event arguments are obtained via independently trained classifiers for each slot, and the final templates are created using the output of these classifiers along with some simple heuristic rules designed to catch common failure cases. The authors experimented with multiple types of classifiers (maximum entropy

[1] While this method does utilize classification techniques, the final extractions are still done via pattern matching, hence why the overall technique falls under the pattern matching category of algorithms

models, support vector machines (SVMs), Naive Bayes, and decision trees), ultimately finding the best performance using the maximum entropy model.

In their evaluation results on the MUC-4 dataset, they found that ALICE was able to outperform nearly all pattern-based methods. While they could not beat the current state of the art model, an important distinction to note is that the top-performing model had required 10.5 person months of effort. In contrast, a learning-based approach takes far less time to develop.

Patwardhan and Riloff [2009] propose a classification-based model that divides the problem into two separate subproblems: plausible role-filler recognition and sentential event recognition. In the plausible role-filler recognition, the goal is to identify candidate event arguments from noun phrases. In the sentential event recognition problem, the goal is to identify the subset of sentences that are focused on a relevant event. By combining these two components together, the hope is to be able to extract event arguments that are central to the event of interest, while avoiding arguments that refer to more generic, off-topic events. In their experiments, they apply Naive Bayes to the plausible role-filler recognition problem, and they consider both Naive Bayes and SVMs for the sentential event recognition problem.

Huang and Riloff [2011] decompose the problem further into a multi-stage pipeline of classifiers. They consider separate classifiers which analyze the text at increasingly fine-grained levels of granularity. Specifically, their model includes document classifiers, event sentence classifiers, role-specific context classifiers, and role filler extractors. SVMs are used for each of these individual classifiers.

Later work by Huang and Riloff [2012] consider the problem from a different perspective. They decompose the problem into two subparts: 1.) a set of independent classifiers which identify argument role fillers, and 2.) a sentence relevance model. The independent classifiers used are identical to the same role filler extractor SVMs used in their previous work [2011]. The sentence relevance model is a conditional random field applied over the sequence of sentences to identify the subset of relevant sentences. The final templates are formed by filtering out any argument role fillers that are not found in relevant sentences.

Recently, neural networks have begun to be considered for use in classification-based approaches to document-level event extraction. Boroş et al. [2014] apply unsupervised neural networks to create word embeddings, which are subsequently used as features for randomized decision trees. However, a key limitation of this method is that it does not incorporate any deep learning techniques for directly training event extraction models, as neural networks are only used for separately learning word embeddings. In their experiments, the authors found this model to outperform all previous baselines.

2.1.2 Sentence-Level Extraction

The task of sentence-level event extraction differs from document-level extraction primarily along the axis of granularity. While document-level extraction assumes the presence of a single primary event, sentence-level extraction can have an unbounded number of events present within the text, each one associated with any number of event arguments. As a consequence of this, there is no notion of a "primary" event in sentence-level extraction. A side-effect of this is that human analysts are unable to determine the importance of different events without reading through the original text themselves.

Historically, the task of sentence-level event extraction originates with the ACE program [32]. A key difference with the types of events that have been studied at the sentence-level compared to document-level research is the generalizability of event types. Event types studied at the sentence-level have focused on more general themes, such as conflicts, transportation of people/items, and life events (see Table 2.2 for a complete list of events considered). This has allowed sentence-level work to capture a much wider range of event types than has been seen under MUC-centric document-level analysis.

Terminology

Let us begin by addressing common terminology seen in the literature for sentence-level event extraction.

- An *event* is something that happens in the world at a particular place and time.
- An *event mention* is a particular occurrence of an event in a document. An event may be mentioned multiple times within the same document, or the same event may be mentioned across a set of documents.
- An *event trigger* is a particular word or phrase that signifies the existence of an event.
- An *event argument* is an entity that fulfills some role within a particular event. The set of valid roles for an event depends on the type of event, including roles such as Agent, Place, and Time.
- An *event argument mention* is a particular textual instance of an event argument.

Sequential Pipelines

The classic approach to sentence-level event extraction is to break the problem down into a pipeline of individual subtasks – namely, trigger identification, trigger classification, argument identification, and argument classification. We describe each of these tasks below:

- Trigger identification – for every word in the document, the system must make a binary prediction as to whether or not the word triggers an event (of any type)
- Trigger classification – given the words that have been identified as event triggers, classify each of them into specific event types (e.g. Attack, Demonstration, etc.)
- Argument identification – given a set of candidate entity mentions (for example, obtained from Named Entity Recognition) and the set of classified event triggers, identify which entity mentions are associated with which events
- Argument classification – given the set of entities associated with each event trigger, classify the relationship within each (entity, event trigger) pair into a specific argument type (e.g. Buyer, Seller, Attacker, Place, Time, etc.)

In some approaches, identification and classification steps are merged into a single classifier, resulting in a two-stage pipeline instead.

Notably, almost all methods for sentence-level event extraction utilize machine learning

methodologies. While early systems utilize some pattern matching for trigger predictions [43][2], the vast majority of systems rely solely on machine learning techniques for classification. This is a vast departure from document-level event extraction, where nearly all early methods relied on handcrafted rules or pattern-matching approaches, with classification only becoming popular in later years.

Structured Prediction

A key problem with using sequential pipelines is that errors can propagate between components. For example, suppose the event trigger identification step failed to detect some particular trigger word. The system will not only suffer a loss on failing to detect the trigger word, but also a loss on any associated event arguments attached to this trigger, as it will become impossible to identify these in later steps.

A pipelined approach can also result in more subtle failure cases. Consider the following sentences:

1. The cannon was fired accidentally, causing three people to receive injuries.
2. The CEO fired his secretary.

The word *fired* is clearly an event trigger in both sentences, but in one case signals a *Conflict.Attack* event, whereas in the other sentence it signals a *Business.End-Position* event. In the first case, analysis of the argument candidate *three people* may suggest that *Conflict.Attack* is the right choice, while in the second case, the argument candidates *CEO* and *secretary* imply that *Business.End-Position* is the event type. However, if event trigger classification is performed prior to considering arguments, the model may mistakenly select the wrong type of event and be unable to correct itself later on when new evidence about the arguments is analyzed.

Another key observation to make about the pipelined model is that it makes argument predictions in isolation of each other. In practice, knowing information about one argument may provide strong evidence for additional argument decisions. Consider the following example sentence:

1. A robber in a Boston bank last night attacked the on-duty security guard, who has since been hospitalized for his injuries.

There are two events in this example, an instance of *Conflict.Attack* and an instance of *Life.Injure*. If we identify that the robber fulfills the *Attacker* role in the *Conflict.Attack* event, this strongly implies that the robber will also fulfill the *Agent* role in the *Life.Injure* event. However, this type of inference requires global knowledge – a pipeline of independent classifiers will be unable to take such evidence into account at prediction time.

These types of failure cases are central to the motivation of the work of Li et al. [2013], who apply a structured perceptron to the task of sentence-level event extraction. The input to the model is formulated as $x = \langle (x_1, x_2, ..., x_s), \mathcal{E} \rangle$, where each x_i is the ith word in the sentence, and $\mathcal{E}$ is the set of argument candidates. The output structure takes the form $y = (t_1, a_{1,1}, ..., a_{1,m}, ..., t_s, a_{s,1}, ..., a_{s,m})$, where t_i is the event trigger assignment to word i and $a_{i,k}$ represents the event argument relationship existing between x_i and argument candidate e_k.

[2]Furthermore, even these methods do not solely use pattern matching, and have some reliance on classification-based approaches

The structured perceptron algorithm operates as follows. At each iteration, the algorithm generates its best assignment to the output variables given the input and its current parameters. If the output correctly matches the gold standard, no additional actions are taken. If not, the model parameters are updated as follows:

$$w = w + f(x, y) - f(x, z)$$

where z is the output produced by the model, and y is the gold standard output according to annotated data.

Subsequent work by Li et al. [2014] expand this model to jointly perform entity extraction, relation extraction, and event extraction in a single model. The overall problem is to learn an information network $y = (V, E)$, where the nodes are either entity mentions or event triggers, and the edges are either extracted relations or event argument roles. The model is learned either by using a structured perceptron (as in their previous work [2013]), or via k-best MIRA, an algorithm for online large-margin learning [86].

Yang and Mitchell [2016] consider the problem of jointly learning event triggers, event arguments, and entity mentions. They model the overall problem as three interconnected subproblems: 1.) within-event structure learning, 2.) event-event relationship learning, and 3.) entity extraction.

Let us begin by considering the within-event structures, which are modeled using the factor-graph representation seen in Figure 2.4. t_i represents the event type of trigger candidate i, $a_1 ... a_m$ represent possible entity candidates in the same sentence as i, and each $r_{i,j}$ represents the argument role type existing between i and a_j. The joint distribution over these variables can be written as:

$$p_\theta(t_i, r_{i.}, a.|i, N_i, x) \propto \exp\big(\theta_1^T f_1(t_i, i, x) + \sum_{j \in N_i} \theta_2^T f_2(r_{i,j}, i, j, x) + \sum_{j \in N_i} \theta_3^T f_3(t_i, r_{i,j}, i, j, x) + \sum_{j \in N_i} \theta_4^T f_4(a_j, j, x) + \sum_{j \in N_i} \theta_5^T f_5(r_{i,j}, a_j, j, x)\big)$$

where $\theta_1 ... \theta_5$ are parameter vectors, $f_1 ... f_5$ are feature extractors, and N_i is the set of entity candidates $a_1 ... a_m$. The model parameters are learned by optimizing the following objective function using the L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) algorithm [78], a quasi-Newton method for optimization:

$$\theta^* = \arg\max_\theta L(\theta) - \lambda ||\theta||_2^2$$
$$L(\theta) = \sum_i \log p_i(t_i, r_{i.}, a.|i, N_i, x)$$

The second subproblem is to consider event-event relationships. The motivation for this is that the presence of one event type may provide evidence either for or against the presence of other event types. For example, Conflict.Attack events are often correlated with the existence of Life.Injure events.

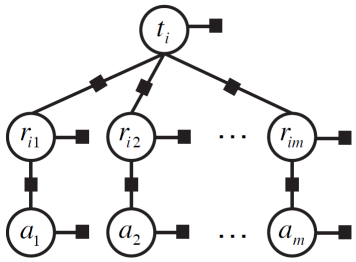


Figure 2.4: Factor graph for within-event structures used by Yang and Mitchell [2016]. t_i represents the event type (e.g. Life.Injure, Conflict.Demonstration) of trigger candidate i. $a_1 ... a_m$ represent the entity candidates occurring within the same sentence as trigger candidate i. $r_{i,j}$ represents the argument role type (if any) existing between i and a_j (e.g. Place, Time, Attacker).

Given a pair of trigger candidates (i, i'), the probability of their event types $(t_i, t_{i'}$ are modeled as:

$$p_\phi(t_i, t_{i'}|x, i, i') \propto \exp\left(\phi^T g(t_i, t_{i'}, x, i, i')\right)$$

where ϕ is a parameter vector and g is a feature function. As with the previous model, parameters are learned using L-BFGS.

The final subproblem is to identify entity candidates, which are used to fill up argument roles. A linear-chain conditional random field is trained to handle entity extraction.

The final task is to combine these models together for joint inference. The model for this is defined in the following manner:

$$\max_{t,r,a} \sum_{i \in T} E(t_i, r_{i.}, a.) + \sum_{(} i, i' \in T) R(t_i, t_{i'}) + \sum_{j \in N} D(a_j)$$

$$E(t_i, r_{i.}, a.) = \log p_\theta(t_i|i, N_i, x) + \sum_{j \in N_i} \log p_\theta(t_i, r_{ij}|i, N_i, x) + \sum_{j \in N_i} \log p_\theta(r_{ij}, a_j|i, N_i, x)$$

$$R(t_i, t_{i'}) = \log p_\phi(t_i, t_{i'}|i, i', x)$$

$$D(a_j) = \log p_\psi(a_j|j, x)$$

where T is the set of trigger candidates, and $p_\psi(a_j|j, x)$ is the marginal probability from the entity extraction CRF.

This problem can be framed as an integer linear program, and is solved using AD^3 (Alternating Directions Dual Decomposition) [85].

Neural Approaches

Nguyen and Grishman [2015] consider the problem of event trigger classification via convolutional neural networks (ConvNets[3]) [62, 66]. For each word in the document, they categorize the

[3]The abbreviation "CNN" is frequently used in the literature to refer to convolutional neural networks, however, this same term can also refer to the broadcast news network. To avoid confusion, we will use the term "ConvNet" to refer to the neural network model, and "CNN" to refer to the news network.

word into one of a set of predefined event trigger types, or "NONE" if the word is not a trigger.

The overall model may be seen in Figure 2.5. At the input layer, the target word x_i and its nearby context words $x_{i-w}...x_{i-1}, x_{i+1}...x_{i+x}$ are each transformed into real-valued vectors using embedding lookup tables for the word, position relative to the target word, and entity mention information. The resulting vectors from each of these tables are concatenated into a single vector, and the set of vectors comprising the target and context words form a matrix x. A convolutional layer is then applied to this matrix, followed by a max pooling layer and fully connected layer. In their experiments, they found that this model was able to slightly outperform existing baselines. However, a notable limitation of this model is that it is unable to detect event arguments.

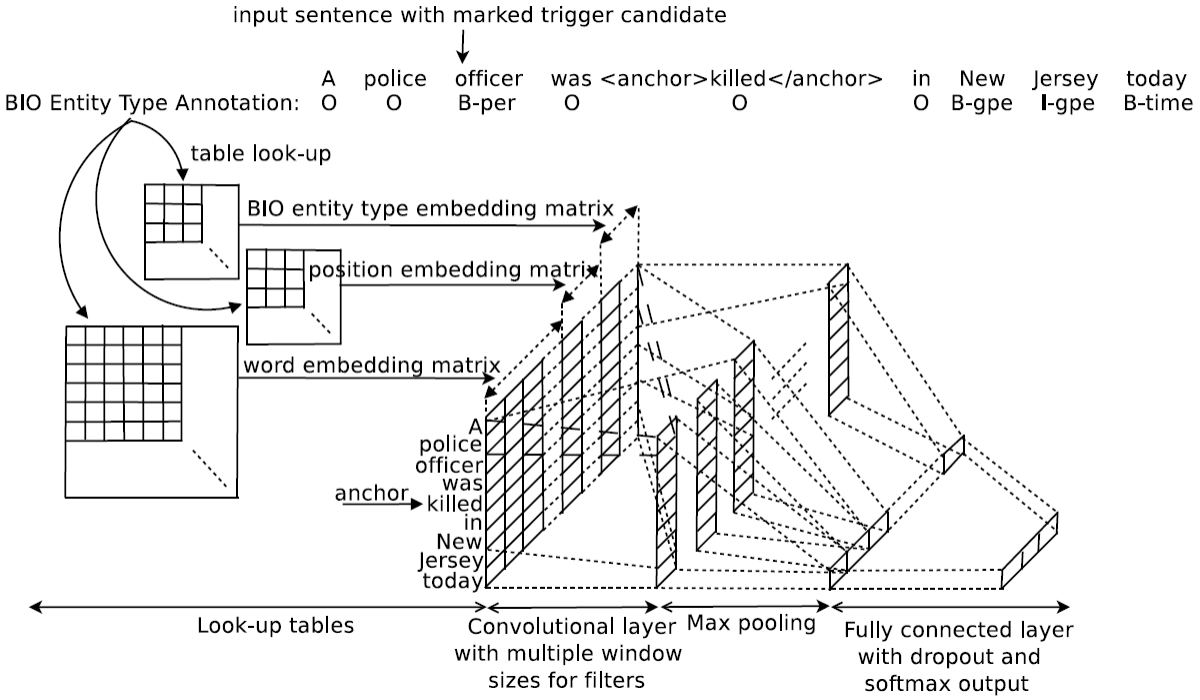


Figure 2.5: ConvNet model used by Nguyen and Grishman [2015]. The convolutional layer allows the model to construct feature maps from words and their contexts. The motivation for the use of ConvNets is to capture information reflecting higher-level semantics, as opposed to solely individual word-level features

Chen et al. [2015] apply a pipeline of convolutional neural networks to first perform event trigger classification, followed by event argument classification. Their overall model architecture, seen in Figure 2.6, is very similar to the model of Nguyen and Grishman [2015]. The below description is the more general model used for argument classification – converting to trigger classification requires only minor modifications to the model.

The model begins with embedding the target and context words into a real-valued matrix. The embeddings consist of lookups for the word, position relative to the target word, and the event trigger output of the word (from the trigger classification network). The resulting matrix is then fed into a dynamic multi-pooling layer, rather than a max pooling layer as seen in Nguyen and Grishman [2015]. The behavior of this layer is almost identical to that of a max pooling

layer, except that instead of collapsing an input feature map into a single value, it instead divides the feature map into three portions and performs a max pooling operation over each (resulting in three output values, rather than one). At the final layer, the dynamic multi-pooling output is combined with the original input word embeddings, and a fully connected layer is applied to perform the final classification of argument roles.

The experimental results for this model show improved performance compared to non-neural methods on both trigger and argument classification. Both neural models perform about the same on event trigger classification.

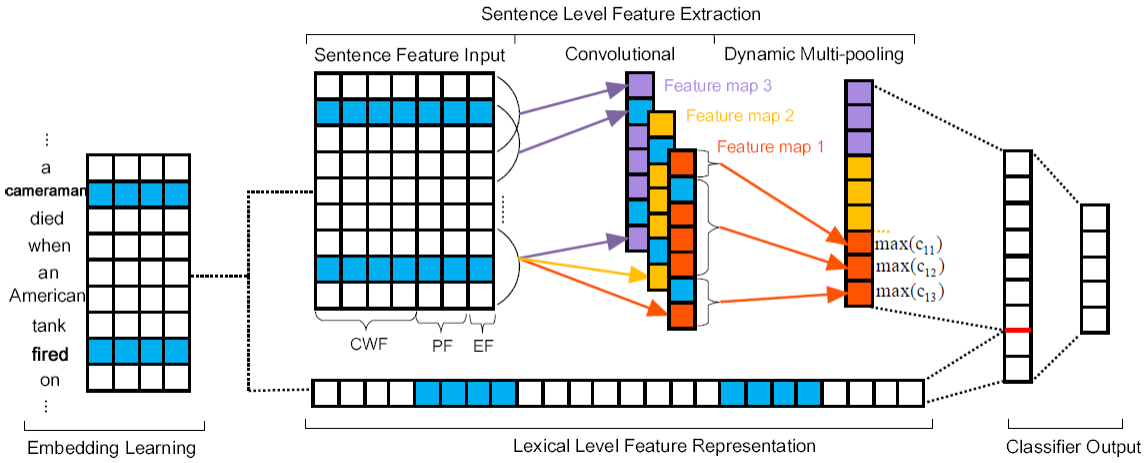


Figure 2.6: ConvNet model used by Chen et al. [2015]. The overall intuition is the same as in the model of Nguyen and Grishman [2015], but is additionally applied to argument extraction, whereas the Nguyen and Grishman model is only used for trigger extraction.

2.1.3 Event Extraction in Non-English Languages

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 [55, 56]. Sentence-level event extraction has been applied to Chinese and Spanish [13, 14, 18, 19, 49, 50, 51, 70]. However, the majority of event extraction work focuses exclusively on English texts.

2.1.4 Event Extraction in Video

In addition to textual event extraction, there exists a large body of work applying event extraction to video [80, 82, 83, 89, 101, 102, 120, 126, 127]. 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 [11, 12, 36, 48, 103, 103, 107]. 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. [44] 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 [45, 93, 100] and more sophisticated deep learning models [16, 28, 31, 60]. Additional details on the field of machine reading comprehension are discussed later in Chapter 6.

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 [38, 39, 40, 59, 79, 88, 98, 110, 111]. Extractive summarization avoids the problem of natural language generation by instead extracting subsets of the original text in order to create a summary [8, 9, 27, 35, 76, 77, 90, 91, 99, 125].

Ultimately however, neither abstractive nor extraction summarization meets the goal of our problem, as both methods simply produce an unstructured natural language text. The unstructured nature of such summaries are often useful for casual users, but are unsuitable for consumption by professional analysts or downstream processing (such as question answering or knowledge base population). 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 [37] extracted relationships between entities and frequent nouns to represent events, and used these as features for extractive summarization. Ji et al. [58] 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.

2.3.1 Extractive Summarization

The extractive summarization problem can be viewed under the following formulation: given a document D containing sentences $(s_1, s_2, ..., s_n)$, select a subset of k sentences that best summarize the text. The most well known method for extractive summarization is Maximum Marginal Relevance (MMR) [8]. MMR operates by greedily selecting sentences one at a time, using a scoring function based on a combination of relevancy and redundancy:

$$\text{score}_{MMR}(s_i) = \lambda \text{rel}(s_i) - (1 - \lambda) \max_{s_j \in S} \text{sim}(s_i, s_j)$$

where λ is a scalar on the interval $[0, 1]$, $\text{rel}(s_i)$ is the relevance of sentence s_i, S is the set of sentences selected so far, and $sim(s_i, s_j)$ is the similarity between sentences s_i and s_j (e.g. cosine similarity). At each iteration of the algorithm, all of the remaining possible sentence candidates are ranked according to their score, and the top sentence is added to the summary. The algorithm continues in this fashion until the desired number of sentences has been obtained.

Another popular approach for extractive summarization is to create a graph of sentences, and apply PageRank [94] to determine the most central subset of sentences. TextRank [91] and LexRank [35] are two popular algorithms based on this idea. In both algorithms an undirected graph $G = (V, E)$ is constructed, where V is the set of sentences, and E is a set of edges

connecting together semantically similar sentences. PageRank is applied to the graph structure, and the overall summary is then constructed from the top scoring sentences.

2.3.2 Abstractive Summarization

Abstractive summarization techniques often begin similarly to extractive summarization techniques, by identifying the key phrases and sentences in the source text that describe the main event. In contrast to extractive summarization however, abstractive summarization does not simply generate summaries by concatenating sentences together, and instead relies on techniques for natural language generation.

McKeown et al. [1999] developed a system which first analyzes the text to identify phrases containing information that will be central to the summary. A parser is applied to the sentences containing these phrases to obtain predicate-argument structures, which are then used by FUF/SURGE [34, 108], a tool for natural language generation. The generated language from FUF/SURGE serves as the final constructed summary.

Jing and McKeown [2000] present a system inspired by "cut and paste" techniques seen in human-generated abstracts. They begin by extracting a set of key sentences which describe the main points of the text, and then edit these sentences down in order to generate a more natural, concise summary. The generation of these summaries is done by using handcrafted rules for sentence reduction and sentence combination. Sentence reduction involves removal of unnecessary phrases from individual sentences, while sentence combination takes information from separate sentences and merges them into a single sentence.

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.

As is typical 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 be-

longs. This corresponds exactly to the notion of an event type in ACE. Example event types that one might want to model include *attacks*, *natural disasters*, *transactions*, *elections*, or *demonstrations*.

The argument roles of the macro-event template are the specific roles that define an event of this type. For instance, if considering an *attack* event, the event could be well-defined by roles of *location*, *time*, *perpetrators*, and *victims*. Additionally, among the victims, we might want to further categorize into *injured victims* and *dead victims*.

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

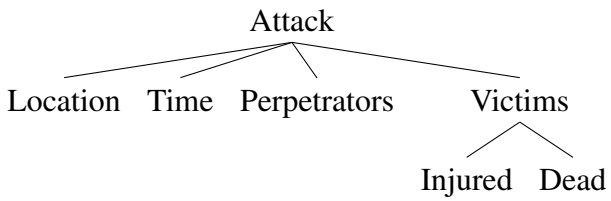


Figure 3.1: Hierarchy defined by the *attack* macro-event type

The hierarchy structure allows for easy definition of constraints among the entity fillers. Sibling relationships between roles in the hierarchy imply mutually-exclusive relationships. Thus, in the *attack* macro-event, any filler that is used in the *victim* slot cannot also be used in the *perpetrator* 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 *attack* macro-event, this means that any *injured victim* must also be included in the more general category *victim*.

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.

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. As an example, consider the following text, and the corresponding macro-event template that would be extracted (Figure 3.2:

- HUNTSVILLE, Texas Condemned killer Johnny Ray Conner asked for forgiveness and said he'd be waiting in heaven for loved ones, including his victim's relatives, as he became the 400th Texas inmate executed since the state resumed carrying out the death penalty a quarter-century ago.

  ...

  He received lethal injection for the slaying of Kathyanna Nguyen, 49, during a failed robbery at her Houston convenience store in 1998. Conner's two sisters were among people

watching through a window as he died. Nguyen's daughter and a sister were among those watching through another window.

...

The *perpetrator* argument is filled using the full name of Johnny Ray Conner as seen in the first sentence of the text. Later mentions of this person in the document use either pronouns or just his surname, but for the purposes of the macro-event it is important to fill the slot with the most identifying form of his name. The *victim* slot is filled by Kathyanna Nguyen, and this same filler is used for the *dead* slot. This reflects the nature of the parent-child relationship amongst argument nodes described previously. The other child node, *injured* remains unfilled, as there were no other victims in this attack. The *time* and *location* slots are similarly filled to include 1998 and Houston respectively.

Attack Macro-Event	
Perpetrators	Johnny Ray Conner
Victims	Kathyanna Nguyen
Victims – Dead	Kathyanna Nguyen
Victims – Injured	(None)
Time	1998
Location	Houston

Table 3.2: An example filled macro-event template for the *attack* domain.

Not all macro-event templates are as simple as this example. Consider the following text and macro-event (Figure 3.3):

- Authorities charged accused Kalamazoo, Michigan, shooter Jason Brian Dalton on Monday with six counts of murder, two counts of assault with intent to commit murder and eight firearms violations.

  ...

  The gunman shot eight people in three different parts of the county Saturday evening, authorities said.

  ...

  Authorities named the first victim as Tiana Carruthers, who was shot in front of her children before 6 p.m. Saturday.

  Next were Richard Smith, 53, and his son Tyler, 17, who were looking at a vehicle at a car dealership when both were shot and killed, police said.

  Tyler's girlfriend, also 17, witnessed the shootings from the back seat of their car, according to Hadley, the Kalamazoo public safety director.

  ...

  The last shooting happened in the parking lot of a Cracker Barrel restaurant. Authorities say four women were killed as they sat in two cars: Dorothy Brown, 74; Barbara Hawthorne, 68; Mary Lou Nye, 62; and Mary Jo Nye, 60.

  A 14-year-old girl who was in the passenger seat of one of the vehicles was also struck. Her family identified her Monday as Abigail Kopf.

  Hadley said Sunday the girl was in "very, very critical condition." A day later, he said the

girl is ”still holding on” and responding to verbal commands.
...

In this example, an individual shooter committed multiple murders in the same evening before being arrested. While sentence-level extraction would treat each attack as a separate event, we instead model the entire evening's attacks via a single macro-event template. In order to correctly fill this template, we need to analyze the entirety of the text. While some information, such as the *perpetrator* and *location* are found very early in the document, the identity and status of the victims is not provided until much later in the text.

Attack Macro-Event	
Perpetrator	Jason Brian Dalton
Victims	Richard Smith Tyler Dorothy Brown Barbara Hawthorne Mary Lou Nye Mary Jo Nye Tiana Carruthers Abigail Kopf
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 3.3: A more complicated *attack* macro-event. This document describes a shooter who killed multiple people in one night, and injured several others. The macro-event structure attempts to summarize the entire attack, rather than treating each individual incident as a separate event (as would be done in sentence-level extraction).

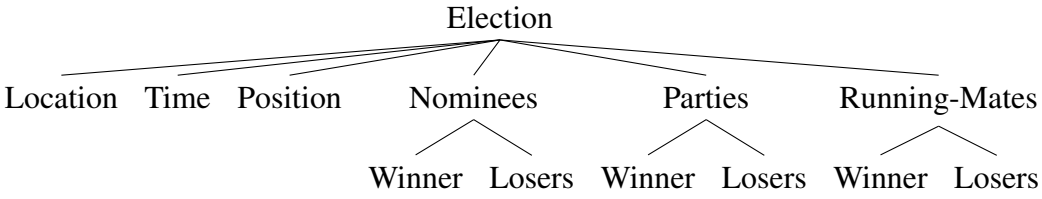


Figure 3.2: Hierarchy defined by the *election* macro-event type

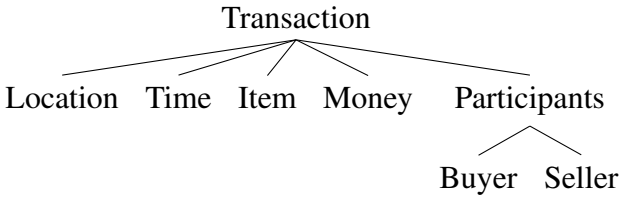


Figure 3.3: Hierarchy defined by the *transaction* macro-event type

3.2 Event Ontology

We have designed structures and collected data for a variety of macro-event types, in order to demonstrate the generalizability of both the macro-event paradigm and our algorithms to a variety of domains. In this section, we will describe each of the studied domains and their respective macro-event templates. In the following section, we will describe the process by which we obtain gold standard annotations for these events.

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.
- Perpetrators – the person(s) instigating the attack
- Victims – the person(s) who received physical harm as a result of the perpetrators' actions
- Injured – the subset of victims who were injured, but not killed as a result of the attack
- Dead – the subset of victims who died as a result of the attack

The hierarchical structure for *attacks* can be seen in Figure 3.1.

3.2.2 Elections

The *election* macro-event consists of the following argument roles[1]:

- 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

[1]For Spanish data, we consider only a subset of these slots, due to lack of available information in Spanish Wikipedia election infoboxes (see Section 3.3.1 for details)

- Parties – the political parties associated with the election
- Party-Winner – the political party of the winner
- Party-Loser – the political parties of the losers
- Running-Mates – the running mates for the election
- Running-Mate-Winner – the winning running mate
- Running-Mates-Loser – the losing running mates

The hierarchical structure for *elections* can be seen in Figure 3.2.

3.2.3 Sporting Events

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
- 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.

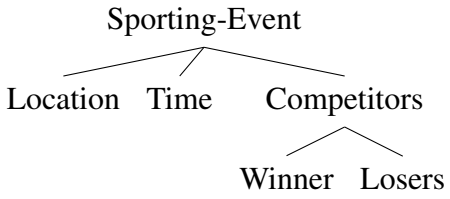


Figure 3.4: Hierarchy defined by the *sporting-event* macro-event type

3.2.4 Criminal Trials

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

- Location – where the trial took place
- Time – when the trial took place
- 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 *criminal-trials* can be seen in Figure 3.5.

One notable difference with the *criminal-trials* macro-event compared to the previously seen macro-event types is the presence of classification-based slots in addition to extraction-based slots. In particular, the *verdict* and *sentence* slots are all classification-based slots, so in these

templates, the macro-event extraction algorithm must select one argument label from a set of predefined possibilities.

For the *verdict* slot, there are three possible argument labels:

- Innocent/Acquittal – the defendant was found not guilty of the crime
- Guilty – the defendant was found guilty of the crime
- Incomplete/Mistrial – the trial is either incomplete, or ended in a mistrial

For the *sentence slot*, there are six possible argument labels:

- Prison – the defendant was sentenced to a prison term
- Execution – the defendant received a death sentence
- Other – the defendant received some sentence other than imprisonment or execution (e.g. community service)
- Unknown – the defendant was found guilty, but the document does not state what the sentence was
- Incomplete (no sentence) – the trial is not complete, and thus no sentence has been given yet
- Acquittal (no sentence) – the defendant was found not guilty, and thus received no sentence

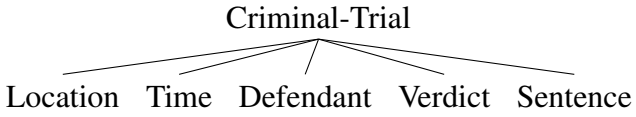


Figure 3.5: Hierarchy defined by the *criminal-trial* macro-event type

3.3 Annotation

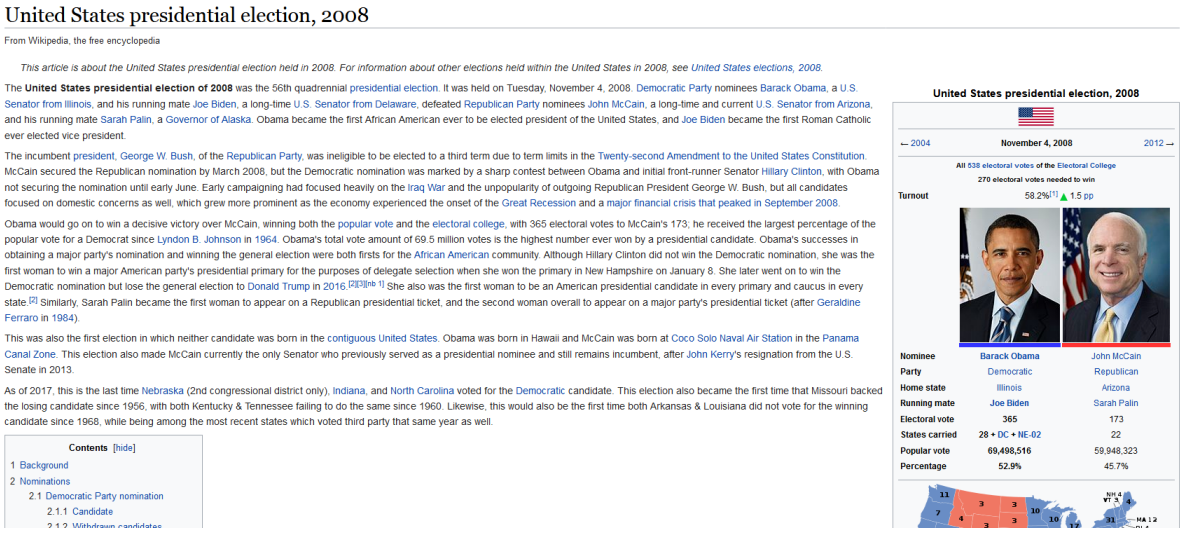
In this section, we will describe the process by which we obtain gold standard annotations for macro-events. We have utilized several techniques for obtaining gold-standard annotation, with the technique used for each particular event type dependent on the availability of existing human-curated resources for the target domain. Corpus statistics for our five datasets may be seen in Table 3.5.

3.3.1 Leveraging Online Resources for Annotation

For some domains, it is possible to leverage existing human-curated 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 [65].



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

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 3.4 for an example macro-event template extracted from an infobox). We applied this process to create macro-event annotated data for *elections* (including both English and Spanish data) and *sporting-events*.

For the *elections* data, we use a different subset of macro-event slots between English and Spanish. This is due to the fact that Wikipedia does not use a consistent schema for infoboxes across languages. As a result, certain slots may be available readily on one language, but not another. For English, we use the full set of slots seen in Figure 3.2. For Spanish, we use all of the slots except the *title* and *running-mate* slots.

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

Table 3.4: A gold-standard macro-event template extracted from the infobox seen in Figure 3.6.

3.3.2 Internal Annotation

For domains without sufficiently detailed existing resources, we have conducted human annotation of documents via Amazon Mechanical Turk[2], a popular service for crowdsourcing work. Specifically, we apply this process to documents for the *attacks* and *criminal-trials* domains. Source documents for our domains are collected from http://murderpedia.org/, a collection of news articles and court documents for famous murder cases.

At annotation time, the user is provided with the full text of a document, and is asked to provide the slot fillers for each of the argument slots belonging to the main event of the text. To ease the burden of annotation, we do not require the annotators to identify entities themselves, and instead provide them with multiple choice questions where the answers are generated via automatic named entity recognition. While in principle this means that some answers may be missed, this has the added benefit of making the task easier for the workers to complete reliably (e.g. avoids the risk of spelling mistakes).

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 is asked to only mark 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 "victims" as well. The annotator is not required to separately identify these slots.

[2] `https://www.mturk.com/`

Corpus Statistics					
	Attacks	Elections (English)	Elections (Spanish)	Sporting-Events	Criminal-Trials
Total # of documents	535	1420	1631	3168	402
Average words per document	434.20	481.85	319.73	468.96	434.8
Maximum words per document	1488	16930	7847	9162	1488
Average fillers per document	4.37	11.8	17.4	5.77	3.19
Maximum fillers per document	12	45	102	7	4
Average fillers per slot	0.73	0.98	2.9	0.96	0.53
Maximum fillers per slot	8	9	42	6	1

Table 3.5: Statistics for macro-event corpora

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 [61, 122]. 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 [109], 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.

Algorithm 1: Learning to Search for Macro-Events Algorithm. π^* is an oracle policy which directly follows the gold standard annotations. We use $\beta_i = I(i = 1)$, as recommended from the work of the AggreVaTe authors.

Data: D, a collection of documents annotated for macro-events
Result: Policy $\hat{\pi}$ for prediction of macro-events on test data

1 Initialization of $\hat{\pi}_1$ to any policy in Π, $X \leftarrow \emptyset$
2 **for** $i = 1$ **to** N **do**
3 Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$
4 Generate k new datapoints using the following procedure:
5 **for** $j = 1$ **to** m **do**
6 Sample a document d uniformly at random from D
7 From d, generate sequence of candidate (entity, argument role) pairs for extraction;
8 Sample a timestep $t \in 1, 2, ...T$, where T is the total number of candidate pairs
9 **for** $k = 1$ **to** $t - 1$ **do**
10 Apply policy π_i to include/exclude candidate pair k from macro-event template
11 **end**
12 Explore all possible policy actions for candidate pair t, for each measuring the total loss obtained via π^* for steps $t + 1...T$.
13 Generate a new training example from the state at timestep t, using the action minimizing total loss
14 **end**
15 Merge new training examples with previous examples: $X \leftarrow X \cup X_i$
16 Train a classifier as the new policy $\hat{\pi}_{i+1}$
17 **end**
18 **return** $\hat{\pi}_N$

The overall algorithm can be seen in Algorithm 1. 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 $\hat{\pi}$) to predict the correct argument roles for each candidate. This can be seen as the first iteration of the loop at line 2 of the algorithm pseudocode. 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 and generate the sequence of candidate pairs (lines 6-7)
2. Follow the current learned policy π_i on this document until a randomly sampled step t

(lines 8-11)

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. (lines 12-13)
4. After generating k new examples, retrain the policy using the new training set, and jump back to step 1. (lines 15-16)

One major point to note regarding this algorithm is that the exploration on each document (lines 6-13) can branch from any point in the sequence of decisions. This allows the algorithm to consider new training examples generated from a wide variety of possible cases.

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 [124].

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 [124]):

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

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.

Chapter 5

Deep Learning for Macro-Event Extraction

Traditionally, neural networks were not widely used models for classification and extraction tasks, due largely to the complexity in training such models. In recent years, however, deep learning has become massively popular for almost all machine learning problems, in part due to major increases in both computing power and availability of large training datasets. In this chapter, we explore the application of deep learning techniques to supervised macro-event extraction. In the following chapter (Chapter 6), we will explore deep learning methods for cases where little to no annotated macro-event training data is available.

5.1 Comparisons to Existing Work

Deep learning has been applied to a wide variety of machine learning and NLP tasks, including part-of-speech tagging [26, 112, 132], parsing [15, 25], named entity recognition [22, 26, 113], machine reading comprehension [16, 28, 31, 44, 60], and computer vision [64, 114, 123] – often resulting in state-of-the-art performance on these tasks.

To date however, deep learning has not been well-studied to the task of document-level event extraction. The closest work to this is that of Boroş et al. [2014]. In their work, they generate word embeddings with a single-layer neural network from unlabeled, on-domain data, and then subsequently use the resulting embeddings as features in an ensemble of randomized decision trees to fill template slots. The goal of this procedure is to automatically learn features that represent the semantic meanings of words, rather than to use human-engineered features.

A key limitation of this method is that no deep learning techniques were actually applied to the main task of event extraction. Neural networks were only used in an unsupervised fashion to obtain word embeddings, and were learned completely separately from the argument prediction classifier. In contrast, in this work we aim to directly solve the task of document-level event extraction with deep learning models, using a multi-layer neural network trained on macro-event annotated data.

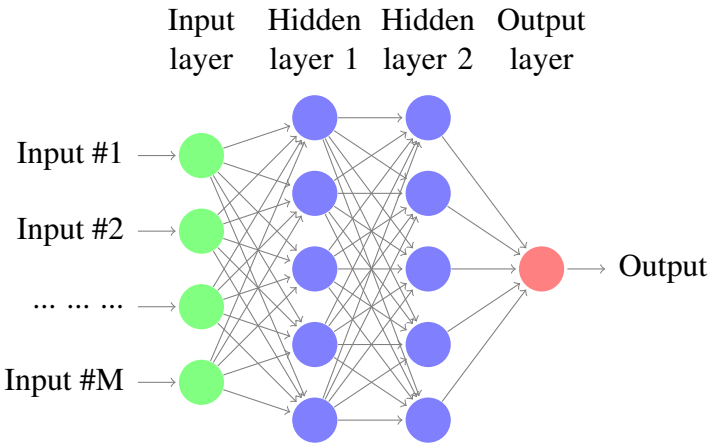


Figure 5.1: Architecture for neural network-based macro-event extraction. Input features are extracted from the document and current *(entity,role)* pair. The input is then fed through multiple hidden layers with ReLU activation functions to add nonlinearity. At the output layer, a binary decision is made to either include or exclude the candidate pair from the macro-event template.

5.2 Model Details

The overall problem we address remains the same as in previous chapters – to fill a document-level event template focusing on the main event in the text. Given an input text and a set of candidate argument fillers, we attempt to fill up a macro-event template by making binary predictions of whether to include or disclude each (entity candidate, argument role) pair in the final template.

Figure 5.1 provides an illustration of our model architecture. The input layer consists of features for our model, extracted in the same manner as seen in the previous chapter (see Table 4.1, local features only). The model includes two hidden layers, followed by an output layer, which provides a probability score for whether the pair should be included in the final template. The motivation for using multiple layers is to allow the model to learn more complex interactions amongst the input features.

Let us consider in more detail exactly how the model operates. Starting at the input layer, we first apply a linear transformation W of the data to a lower-dimensional hidden vector (where the weights of W are parameters to be learned).

After applying this transformation, the hidden vector is then passed through a non-linear transformation, which allows the network to model more complex decision boundaries. Popular choices for non-linear transformations include the sigmoid, hyperbolic tangent, and rectified linear unit (ReLU). The equations for each of these are seen below (corresponding graphs may be seen in Figures 5.2, 5.3, and 5.4.

$$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{5.1}$$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{5.2}$$

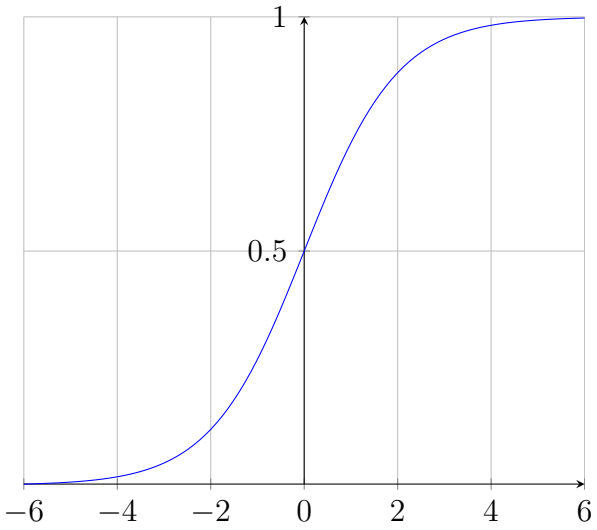


Figure 5.2: Visualization of the sigmoid function

$$\text{ReLU}(x) = \max(0, x) \tag{5.3}$$

In our experiments, we use rectified linear units, which gave the best performance on validation.

This process is subsequently repeated for the additional layers, until we reach the final layer. At the final layer, we apply a softmax transformation over the data, which converts the scores for each possible output decision into a set of probabilities:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{5.4}$$

Algorithm 2 shows the process by which we train our model parameters. The overall process occurs over k epoches. Each epoch involves the following steps. We first apply the model to each individual training example. For each training example, we run the data through the network to obtain a prediction using the current parameters. We then compare this to the true label, and run backwards through our network, updating each parameter using the backpropagated gradient. Once the parameters have been updated, the model moves on to the next training example.

After the model has been applied to all of the training examples, we evaluate the current model using a held-out set of validation data. If the current model parameters provide the highest performance seen so far on the validation parameters, we store these parameters as the new best parameters. At the end of the k epoches, the parameters that provided the best validation performance are returned, and applied to the test data. Selection of a value for k is dependent on much one wants to balance between efficiency and model performance. Too small of a value for k may result in suboptimal performance, while too large of a value for k will lead to a training process that is much longer than necessary[1]. In our experiments, we found that $k = 10$ gave

[1] After a certain number of iterations, the model is likely to start overfitting the training data, hence there is little to be gained from subsequent epochs.

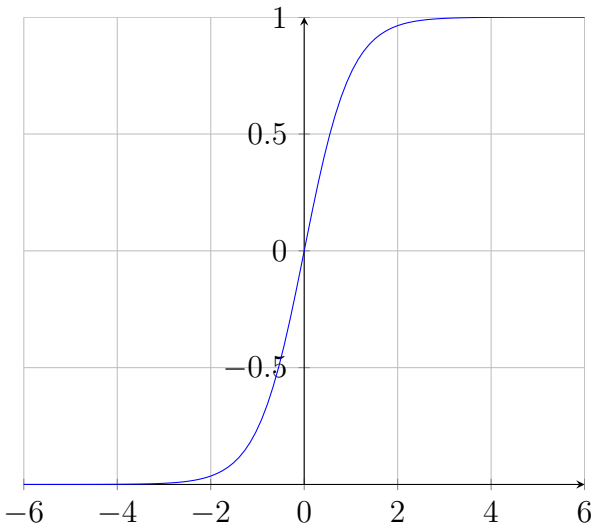


Figure 5.3: Visualization of the hyperbolic tangent function

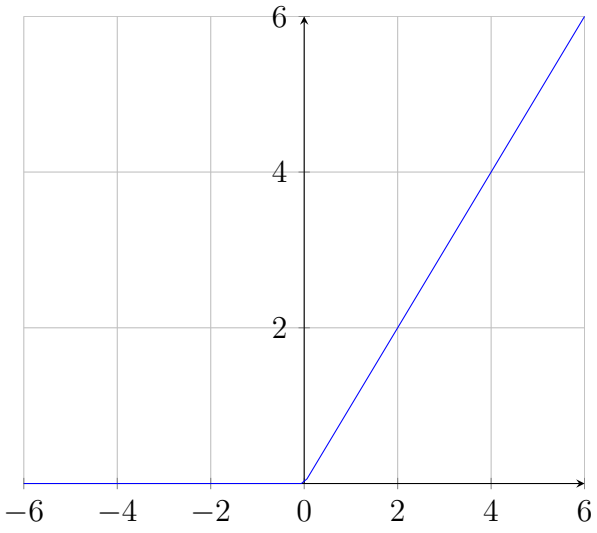


Figure 5.4: Visualization of a rectified linear unit (ReLU)

good performance.

Algorithm 2: Training procedure for our model in Figure 5.1. The model is trained over k epoches. In each epoch, a prediction is made for each training example given the current parameters, and the weights are updated via backpropagation of the loss. After all training examples have been visited, we evaluate the current model using validation data. At the end of the algorithm, the parameters with best performance on the validation set are returned.

Data: $X = x_1, x_2, ...x_N$ training examples, $Y = y_1, y_2, ...y_N$ training labels,
$V = v_1, v_2...v_M$ validation examples, $Z = z_1, z_2...z_M$ validation labels
Result: Model parameters W for neural network

Initialization of W using random samples from uniform distribution, $F1_{best} \leftarrow 0$;
for $i = 1$ **to** k **do**
  for $j = 1$ **to** N **do**
    $loss \leftarrow L(f(x_j, W), y_j)$
    $W \leftarrow backpropagation(loss, W)$
  end
  $F1 \leftarrow eval(V, Z, W)$
  if $F1 > F1_{best}$ **then**
    $W_{best} \leftarrow W$
    $F1_{best} \leftarrow F1$
  end
end
return W_{best}

The overall optimization method used for this process is stochastic gradient descent. An important hyperparameter in the model is the learning rate for the optimization. Too large of a learning rate causes the model to vary wildly from iteration to iteration, as it overshoots desired local minima. Too small of a learning rate can cause the model to converge slowly, and also runs the risk of getting stuck in local minima, rather than reaching the global minimum. In our experiments, we tune our learning rate using the validation set.

We additionally apply dropout to the input layer during training [47]. Dropout is a popular technique for training deep learning models, where a random number of neurons are set to zero during each round of training. The purpose of this is to avoid overfitting the model to the training data, and can be seen as a form of regularization. At test time, no dropout is applied to the network.

Chapter 6

Neural Machine Reading Comprehension

A key disadvantage of both rule-based methods and learned classifiers is that significant human effort is required to deploy these methods to any new domain. For rule-based methods, this involves time to design and curate rules and patterns for extraction; for classification-based methods, this means large-scale annotation of documents. Both are undesirable for rapid deployment to a new domain of interest.

In this chapter, we propose a novel algorithm that avoids the costs associated with both rule-based and machine learning approaches to event extraction. In particular, we introduce a method which leverages 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. The overall time to deploy such a model to a new event-extraction domain is vastly reduced compared to the substantial human effort required by past approaches to event extraction.

Central to our approach is the Gated-Attention (GA) reader [31]. The GA reader is a deep learning model that uses a multi-hop architecture combined with an attention mechanism, achieving high performance 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.

6.1 Model Details

Figure 6.1 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. The motivation for using k layers is to allow the model to build up a more complex representation of the input document tokens, with each layer providing focused attention on a different aspect of the query. Layer-specific document and query embeddings are each independently transformed using bi-directional Gated Recurrent Units (GRU) [23], and then combined using a Gated-Attention module. The Gated-Attention module is applied to each word embedding d_i in

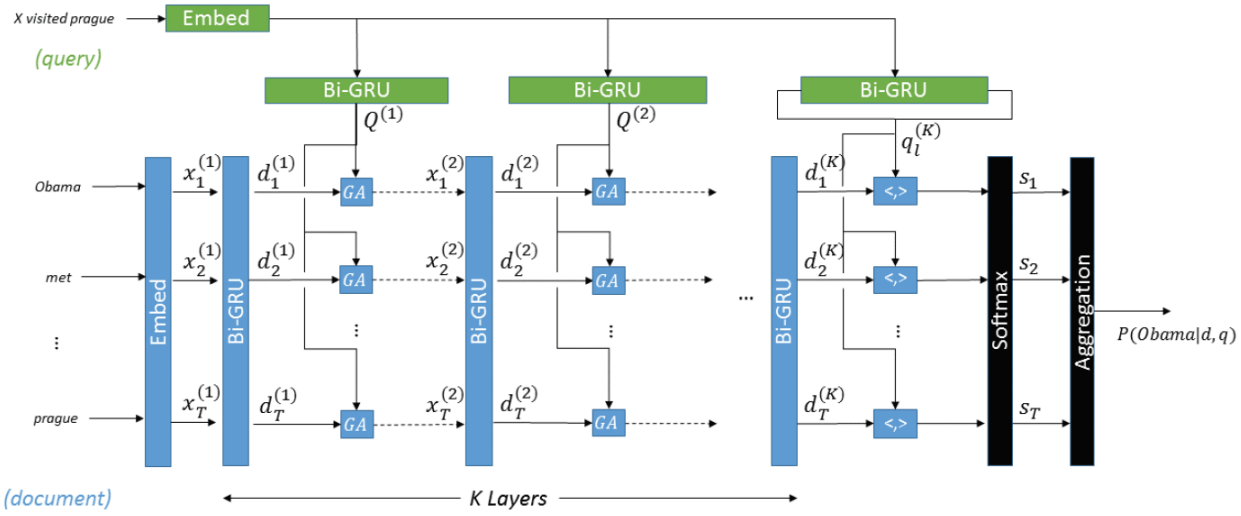


Figure 6.1: Architecture for the GA Reader. Embeddings for both the query and document words are obtained via look-up tables, and then passed through bi-directional GRUs to obtain a transformed representation. Each "GA" box represents a Gated-Attention module, which apply query-focused attention to the document representation. After repeating this process over multiple layers, a score is computed for each word in the document. These scores are converted to a probability distribution over the words, and probabilities for repeated words in the text are aggregated. The final resulting probability distribution is used to select answers for the query.

the document, in the following manner:

$$\begin{aligned} \alpha_i &= \text{softmax}(Q^T d_i) \\ \tilde{q}_i &= Q\alpha_i \\ x_i &= d_i \odot \tilde{q}_i \end{aligned} \tag{6.1}$$

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:

$$\begin{aligned} \Pr(c|d,q) &\propto \sum_{i \in \mathbb{I}(c,d)} s_i \\ a^* &= \arg\max_{c \in C} \Pr(c|d,q) \end{aligned} \tag{6.2}$$

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.

6.2 Training Data

A variety of large-scale corpora for machine reading comprehension have been developed in recent years, including the CNN/DailyMail [44], CBT (Children's Book Test) [46], SQuAD (Stanford Question Answering Dataset) [100], and WDW (Who Did What) datasets [93]. 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.

Let us briefly describe some popular machine comprehension datasets studied in the literature.

6.2.1 CNN/DailyMail

The CNN and DailyMail datasets were the first major large-scale datasets for training machine comprehension model. Prior to their development, previous corpora were at a scale of merely hundreds of examples, and thus could not be used to train data-intensive models.

The approach for creating this dataset was quite simple. First, a large number of articles were collected from the CNN and DailyMail websites (93k and 220k respectively from each source). Of key importance is that articles from both websites contain human-generated bullet-point summaries of the documents, which serve as short, abstractive summaries of the texts. For each bullet point, a (document, query, answer) tuple was generated by replacing one randomly selected entity with a placeholder. An example of this may be seen in Figure 6.2.

6.2.2 CBT

Similarly to the CNN/DailyMail datasets, the CBT dataset is generated automatically from texts without reliance on human annotation. In this dataset, the source documents come from publicly available children's books.

(document, query, answer) pairs are generated using the following procedure. For each chapter in a book, the first 20 sentences are used as the source document. The question and answer are created by taking the 21st sentence, and randomly replacing one word with a placeholder.

Beyond the true answer, additional candidate answers are generated by selecting words at random from the source document. The randomly selected words must belong to the same category of word as the true answer – one of Named Entities, Common Nouns, Verbs, or Prepositions. These categorizations are obtained by applying automatic tools for part-of-speech tagging and named entity recognition.

6.2.3 SQuAD

The SQuAD dataset is another well-known, popular dataset for machine reading comprehension. Unlike the CNN/DailyMail datasets, SQuAD was generating using crowdsourced human answers, rather than replacement of entities in bullet point summaries.

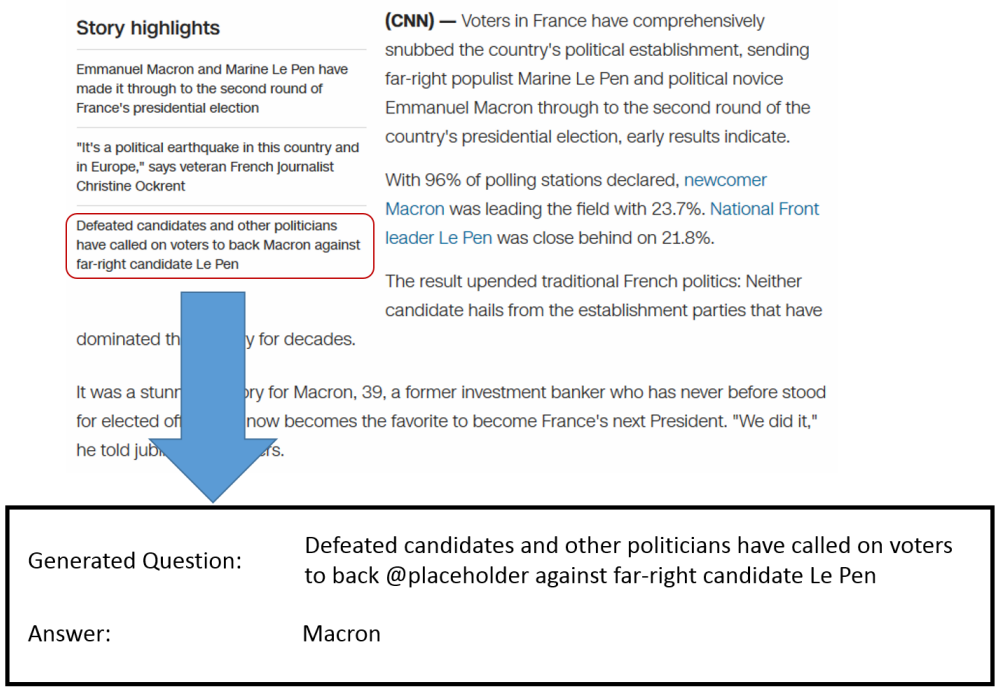


Figure 6.2: Process by which (document, question, answer) tuples are generated for CNN data.

First, passages from Wikipedia were obtained for use as the source documents in the dataset. From a collection of 10,000 articles with highest PageRank scores, 536 articles were sampled randomly. The articles were filtered to remove information such as graphs, tables, or figures, and split into individual paragraphs.

The resulting 23,215 paragraphs were then provided to crowdsourced workers, who were asked to create and answer questions based on the content of their assigned paragraph. To facilitate the creation of high quality question/answer pairs, workers were provided with an example paragraph and samples of both good and poor question/answer pairs. The resulting question/answer pairs from this crowdsourcing make up the datapoints in the SQuAD dataset.

6.2.4 WDW

The WDW dataset, similarly to the CNN/DailyMail datasets, does not rely on using human annotation to produce (document, query, answer) tuples. Unlike these datasets, however, the process for creating these tuples does not rely on using bullet point summaries of articles, instead choosing to use two separate articles that both focus on the same event. A key advantage to this approach is that is enables the creation of datasets for machine reading comprehension which may not have human-generated summaries available.

The overall process for creating the WDW dataset is as follows. First, an article is randomly sampled from the Gigaword corpus, and the first sentence is selected for use as the source of the question. From this sentence, one named entity is randomly selected, and the entire noun phrase containing this entity is removed with a placeholder.

Once the question is obtained, the only remaining step is to obtain a corresponding passage for use as the source document. A source document is obtained from Gigaword with an information retrieval system, using the first sentence from the original document as the query.[1]

6.2.5 Comparison of Datasets

Information on each of these datasets can be seen in Table 6.1. The largest of these is the DailyMail dataset, at nearly 1 million questions. However, none of the datasets can be considered small – even the smallest dataset (SQuAD) consists of more than 100k example questions.

Almost all of the datasets avoid the issue of human annotation by creating questions through entity replacement. Only the SQuAD dataset uses actual human annotation (of both questions and answers).

One of the most important considerations when comparing the datasets is the type of questions included in the data. CBT only includes questions generated from children's books, which is a very limited domain. As such it is not well-suited to our domains of interest. SQuAD consists of Wikipedia articles, which allows it to cover a broad range of topics, ranging from music to history to mathematics. However, as our task is focused specifically on news, the broad coverage of this data is not necessarily an advantage, as this means fewer questions are likely to be relevant to our target domains.

The remaining three datasets (CNN, DailyMail, WDW) all consist of questions generated from news stories. This domain is a strong match for our domains of interest, and models trained on such data would be well-equipped to handle macro-event extraction. Of these models, we select the WDW dataset for our experiments. There are two main reasons for selecting this dataset over either CNN or DailyMail. The first is the difficulty of the questions among these datasets. The questions in WDW are generally considered to be more challenging, in that they require more semantic analysis to properly solve. In contrast the CNN and DailyMail datasets have much closer syntactic similarity between the questions and source documents, making the learned models focus more on sentence similarity than semantic relatedness. The second reason is that the CNN and DailyMail datasets artificially increase the difficulty of problems by anonymizing named entities in the questions and documents. While this could be argued to make the problem more interesting from a machine learning perspective, as it prevents systems from using background knowledge about named entities, in practice it would not be beneficial for a real-world system to limit itself in such a manner. With our goal of macro-event extraction, there is no reason to block the use of background knowledge in our system.

6.3 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

[1] Some restrictions are applied during the search – for example, the new document must have a publication date within 2 weeks of the original document, and the named entity answer must be contained within the selected article.

Dataset	Number of Questions	Source of Documents	Human Annotation Required
CNN	387420	News	No
DailyMail	997467	News	No
CBT	687343	Children's Books	No
SQuAD	107785	Wikipedia Articles	Yes
WDW	147786	News	No

Table 6.1: Information on each of the major datasets for machine reading comprehension.

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 6.2.

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

Chapter 7

Experimental Results

In this section, we will describe our experimental results for macro-event extraction. We apply macro-event extraction experiments to the following domains: *attacks*, *elections*, *sporting-events*, and *criminal-trials*. (See Figures 3.2, 3.1, 3.4, and 3.5 for their respective hierarchical structures). Furthermore, for the *elections* domain, we report results on both English language documents and Spanish language documents. We use the same macro-event datasets described in Chapter 3.

7.1 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 [71, 72, 73]. We directly use the implementation provided via the RPI Joint Information Extraction System[1]. To match our annotation scheme, we only consider arguments that are named entities.
- QA Neural Networks – our proposed neural machine reading comprehension model for macro-events described in Chapter 6
- Independent SVMs – linear SVMs using using the local features described in Chapter 4
- Learning to Search for Macro-Events – our proposed model described in Chapter 4
- Independent Neural Networks — our proposed deep learning model trained directly on macro-event training data, described in Chapter 5

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

	Supervised?	Requires On-Domain Training Data?
First Mention Baseline	No	No
Most Frequent Baseline	No	No
Aggregated Sentence-Level	Yes	Yes
QA Neural Networks	Yes	No
Independent SVMs	Yes	Yes
Learning to Search for Macro-Events	Yes	Yes
Independent Neural Networks	Yes	Yes

Table 7.1: Training data requirements for macro-event extraction algorithms

7.2 Experimental Setup

For each dataset, we randomly partition the data into separate training, validation, and testing sets. Preprocessing of each document is done using Stanford CoreNLP[2] [84] to obtain POS tags, named entities, entity coreference, and dependency parsing for our features.

Different methods have different requirements for training data, as seen in Table 7.1. The first mention and most frequent baselines do not require any training data. All other methods require available training data, but differ in the type of training data required. The aggregated sentence-level event extraction approach requires sentence-level annotated data for all target domains, and is trained using the ACE 2005 data for our experiments. The QA Neural Network does not require any on-domain training data for target events, and can be trained using any large machine comprehension corpus. In our experiments, we train our model using the WDW data. Lastly, the independent SVMs, Learning to Search, and independent neural network methods all require on-domain macro-event annotated data, and are trained using the training set from our annotated data.

Our independent SVMs and Learning to Search methods are both trained using LIBSVM[3][4] [10]. Our independent neural network approach is implemented using PyTorch[5].

For the aggregated sentence-level approach, we combine arguments from multiple types of sentence-level events based on the domain. For the *attack* dataset, we combine argument results from the *Conflict.Attack*, *Life.Injure*, and *Life.Die* event types into a single event template. For the *elections* dataset, we combine argument results from the *Personnel.Elect* and *Personnel.Nominate* event types. For the *criminal-trials* domain, we combine argument results from the *Justice.Trial-Hearing*, *Justice.Charge-Indict*, *Justice.Convict*, *Justice.Sentence*, and *Justice.Acquit* events. We do not apply this method to the *sporting-events* domain, due to lack of sentence-level annotations, nor do we apply this method to the Spanish *elections* data, as the RPI Joint Information Extraction System does not include functionality for Spanish extraction.

Parameters for all methods are tuned using the held out validation set. We evaluate all models using micro and macro-averaged F1 (the harmonic average of precision and recall).

[2]https://stanfordnlp.github.io/CoreNLP/

[3]https://www.csie.ntu.edu.tw/ cjlin/libsvm/

[4]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.

[5]http://pytorch.org/

7.3 Results

Let us begin by considering the experimental results on each individual dataset. We will then proceed to discuss general trends observed over these experiments

7.3.1 Attacks

On the *attacks* domain, we find that the aggregated sentence-level method performs the worst by far. This performance clearly indicates that sentence-level methods are insufficient to solve the macro-event extraction task. The first mention and most frequent baselines are the next lowest performers on both macro-averaged and micro-averaged F1. QA Neural Networks shows a substantial increase in performance over these baselines, with a 41.6% increase in micro-averaged F1 over the most frequent baseline, and a 28.2% increase in macro-averaged F1 over the most frequent baseline. The methods trained directly on macro-event data are the strongest performers, with even the worst methods showing 26.1% improvement on micro-averaged F1 and 18.5% improvement on macro-averaged F1 compared to the QA Neural Networks. On micro-averaged F1, independent SVMs and independent neural networks perform at about the same level, with Learning to Search showing an additional 8.3% improvement over the next best method. On macro-averaged F1, independent neural networks and Learning to Search perform approximately the same, with independent SVMs lagging slightly behind both.

	Micro-Averaged			Macro-Averaged		
	Precision	Recall	F1	Precision	Recall	F1
First Mention Baseline	24.2	26.6	25.9	25.3	24.7	24.7
Most Frequent Baseline	26.2	29.1	27.6	26.3	27.7	26.2
Aggregated Sentence-Level	5.3	3.5	4.2	4.8	3.6	3.9
QA Neural Networks	28.0	64.7	39.1	25.6	52.8	33.6
Independent SVMs	54.8	44.9	49.4	44.4	36.6	39.8*
Learning to Search for Macro-Events	60.1	48.2	**53.5***	45.8	37.3	**40.7***
Independent Neural Networks	54.7	44.8	49.3	45.2	38.0	40.6*

Table 7.2: Results from macro-event extraction on the *attacks* domain. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method.

7.3.2 Sporting-Events

On *sporting-events*, we find the first mention baseline to be the lowest performer. This time, however, there is a substantial difference between first mention performance and most frequent performance, with most frequent showing a 53.4% increase on micro-averaged F1, and a 62.8% increase on macro-averaged F1. These results indicate that on this domain, slot fillers are much more likely to be mentioned repeatedly in the document compared to results on the *attack* domain.

QA Neural Networks perform only slightly better than the most frequent baseline on macro-averaged F1, and slightly worse on micro-averaged F1. The reason for the weaker performance on this data compared to the stronger results on *attacks* is likely due to the fact that the *sporting-events* domain focuses primarily on ORGANIZATION named entities, rather than PERSON named entities. This creates a larger gap between the target domain and the source dataset, which focuses mainly on people and their actions.

The next best method is the independent SVMs approach, which is the weakest of the three methods directly trained on macro-event annotated data. Compared to the next best approach, we see a massive increase in performance even with just independent SVMs – 104.7% improvement on micro-averaged F1 and 93.3% improvement on macro-averaged F1. The reason for this substantially larger improvement compared to the *attacks* domain is due to the much larger pool of annotated training data available for the *sporting-events domain*.

On both micro-averaged F1 and macro-averaged F1, we find that the independent neural networks approach is the top performing method, with Learning to Search performing slightly behind it.

	Micro-Averaged			Macro-Averaged		
	Precision	Recall	F1	Precision	Recall	F1
First Mention Baseline	20.7	27.5	23.6	19.6	25.4	21.5
Most Frequent Baseline	30.3	45.0	36.2	30.7	43.6	35.0
QA Neural Networks	24.2	68.5	35.7	28.5	66.9	37.4
Independent SVMs	74.4	73.9	74.1	72.4	72.4	72.3*
Learning to Search for Macro-Events	76.8	72.0	74.3	75.7	71.2	73.2*
Independent Neural Networks	80.4	73.0	**76.5***	78.4	70.1	**73.4***

Table 7.3: Results from macro-event extraction on the *sporting-events* domain. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method.

7.3.3 Criminal-Trials

The *criminal-trials* domain differs from the other macro-event types in that it includes classification-based slots in addition to extraction-based slots. For those methods which do not utilize annotated macro-event training data (e.g. first mention baseline), classification slots are filled using a baseline of randomly selecting among the possible answers.

The worst performing method for *criminal-trials* is the aggregated sentence-level method, which is far lower than any of the competing approaches. The next lowest performance is seen with QA Neural Networks, followed by the first mention and most frequent baselines. A key factor for the comparatively high performance of the heuristic methods is that the *criminal-trials* domain has a much lower number of fillers per slot compared to other domains. On datasets with higher numbers of argument fillers, the first mention and most frequent baselines typically undergenerate on extracted arguments, rendering these methods less competitive against methods that can choose a dynamic number of argument fillers for any given input.

The methods trained directly on macro-event data all once again show superior performance compared to the previous algorithms. The weakest of these (independent SVMs) shows a 65.5% improvement over the most frequent baseline on micro-averaged F1, and an improvement of 25.7% over the same baseline on macro-averaged F1. The best method for micro-averaged F1 is the Learning to Search method, followed by independent neural networks. On macro-averaged F1, the best method is the independent neural networks, with Learning to Search coming in second place.

	Micro-Averaged			Macro-Averaged		
	Precision	Recall	F1	Precision	Recall	F1
First Mention Baseline	31.9	37.8	34.6	31.5	37.4	33.8
Most Frequent Baseline	32.5	40.0	35.9	32.4	50.2	35.4
Aggregated Sentence-Level	18.3	14.0	15.8	11.8	12.1	11.9
QA Neural Networks	32.6	34.7	33.6	30.3	31.9	31.0
Independent SVMs	70.9	51.1	59.4	46.9	42.8	44.5
Learning to Search for Macro-Events	70.4	54.1	**61.2***	49.7	45.8	47.3
Independent Neural Networks	73.5	50.5	59.8*	63.4	44.6	**51.8***

Table 7.4: Results from macro-event extraction on the *criminal-trials* domain. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method.

7.3.4 Elections

Let us begin first by considering performance on the English election documents. As seen on the other datasets, the aggregated sentence-level method performs far worse than any other method. On micro-averaged F1, the heuristic methods are the next worst, as usual. QA Neural Networks show a very large improvement (53.5%) over these methods, and independent SVMs shows a further 51.5% improvement over the QA Neural Networks. Independent neural networks perform slightly better than independent SVMs, and Learning to Search provides an even further gain.

On macro-averaged F1, the situation is slightly different. The bottommost method (apart from the abysmal sentence-level approach) is the QA Neural Networks, followed closely by the heuristic methods. The main reason for this is because the QA Neural Networks are strong on the argument roles with many fillers, but very weak on the rare argument roles – namely the ones focused on either political parties or running-mates. For political parties, the low performance results from the same issue as in the *sporting-events* domain – the source-training data is not as well equipped to handle ORGANIZATION named entities as compared to PERSON named entities. On running-mates, the model has difficulty distinguishing running-mates from the primary candidates, who are typically the more central focus of election-themed documents.

Performance from the remaining methods follows similar patterns to the previous datasets. Independent SVMs show a 78.5% improvement compared to the first mention baseline, and independent neural networks show a slight improvement beyond that. The top-performing method

for macro-averaged F1 is the Learning to Search method, showing a 8.1% improvement over the next best method.

Now, let us consider performance on Spanish texts. On Spanish, the most frequent baseline is the bottommost performer, at 29.8 micro-averaged F1 and 28.4 macro-averaged F1. The first mention baseline provides a slight improvement of 6.0% on micro-averaged F1 and 5.3% on macro-averaged F1. On micro-averaged F1, the next best performer is the independent SVMs method, followed by the independent neural networks. On macro-averaged F1, the situation is flipped, with independent SVMs showing a stronger performance than independent neural networks. The top performing method for both micro-averaged and macro-averaged F1 is the Learning to Search approach, which shows an improvement of 1.9% on micro-averaged F1 and an improvement of 5.0% on macro-averaged F1 over the next best scores for each.

Note that we do not apply the QA Neural Networks to the Spanish election documents, as the source training data for the method does not contain Spanish data.

	Micro-Averaged			Macro-Averaged		
	Precision	Recall	F1	Precision	Recall	F1
First Mention Baseline	21.0	41.6	27.9	21.8	27.0	22.2
Most Frequent Baseline	22.8	48.7	31.0	22.7	33.4	25.1
Aggregated Sentence-Level	63.2	11.4	19.3	16.3	8.1	9.9
QA Neural Networks	43.5	52.5	47.6	17.2	30.0	20.7
Independent SVMs	78.5	66.7	72.1	52.9	41.7	44.8
Learning to Search for Macro-Events	75.2	73.1	**74.1***	50.6	49.7	**49.6***
Independent Neural Networks	81.7	67.0	73.6*	52.7	43.3	45.9

Table 7.5: Results from macro-event extraction on the *elections* domain, English documents. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method.

	Micro-Averaged			Macro-Averaged		
	Precision	Recall	F1	Precision	Recall	F1
First Mention Baseline	27.4	37.4	31.6	27.7	35.6	29.9
Most Frequent Baseline	25.8	35.3	29.8	26.0	34.6	28.4
Independent SVMs	54.0	46.9	50.2	50.1	44.1	46.3
Learning to Search for Macro-Events	55.7	50.0	**52.7***	50.9	48.3	**48.6***
Independent Neural Networks	61.1	44.8	51.7*	61.1	40.6	45.1

Table 7.6: Results from macro-event extraction on the *elections* domain, Spanish documents. Numbers in bold represent the best F1 performance among all methods. Entries with * represent no significant difference (at $\alpha = 0.05$) compared to the best method.

7.4 Discussion of Results

Across all datasets, we consistently find that the aggregated sentence-level approach performs poorly. Although it is not surprising that a sentence-level focus would not be optimal for a document-level task, it is surprising to see that this method can be easily beaten by even simple heuristic approaches.

We find that the heuristic methods are typically the next lowest performing methods on both micro-averaged and macro-averaged F1. This is not surprising given the simplicity of these models. QA Neural Networks typically perform somewhere in between the heuristic methods and the models trained directly on macro-event annotated data. This indicates that for domains with sufficient training data, it is better to directly train a macro-event extractor than to utilize a model trained on off-domain data.

Among the three algorithms that are trained directly on macro-event annotated data, the independent SVMs are clearly the weakest. This method fails to achieve the top performance for any macro-event domain on either micro-averaged or macro-averaged F1.

On micro-averaged F1, the Learning to Search method performs strongest on all domains except for *criminal-trials*. On macro-averaged F1, the Learning to Search method performs stronger on *attacks* and *elections*, while the independent neural networks perform stronger on *criminal-trials* and *sporting-events*.

7.5 Error Analysis

Let us now consider more detailed analysis of results via the *elections* domain. Figure 7.1 shows the distribution of training examples over the 12 different slots contained within the *election* macro-event type. Some argument types, such as *nominee* and *winner* have many training examples. Other argument types have far fewer examples, as is the case with the *party* and *running-mate* slots.

As one would expect, the availability of training data can be a limiting factor for argument detection. The most extreme case of this is seen in the *running-mate*, *running-mate-winner*, and *running-mate-loser* slots. None of the top performing models are able to perform accurate extractions for these slots. Similarly, we see that the next rarest slots (*party-winner* and *party-loser*) also have noticeably lower scores than other slots, albeit much higher than the aforementioned *running-mate* slots. In contrast, the *nominee* slot, which has by far the most training examples, consistently sees F1 performance in the mid to upper 70s across all three methods.

However, training data alone is clearly not the only factor involved in determining the performance of our models on rare categories. Consider the *time* and *location* slots, which achieve substantially higher F1 scores than *nominee*, despite having hundreds fewer of examples. The *time* slot reaches F1 scores into the mid-80s, while the *location* slot does even better with F1 scores into the 90s. By comparison, the *nominee* slot only can achieve an F1 in the mid-70s. The reason for these slots achieving such high scores despite having less training data is due to the semantic difficulty of the slot itself. Times and locations for elections are typically very easy to identify within a Wikipedia article, often occurring within the first paragraph or even the first sentence. In contrast, discussion of the nominees may be delayed until later in the document,

particularly for nominees who did not win the election, who in general receive less focus than the winners.

One particular strength of the Learning to Search approach can be seen in its ability to extract arguments that are child nodes of another argument. Compared to the independent SVMs method, the Learning to Search approach is consistently able to achieve higher F1 scores for argument slots that belong to a parent-child relationship (with the exception of the *running-mate* slots – as discussed previously, none of the top models are able to extract well for these slots). Among the *party*, *party-winner*, and *party-loser* slots, Learning to Search shows far better performance than independent SVMs, especially in the cases of the *party-winner* and *party-loser* slots, which have far fewer training examples. Among the *nominee*, *winner*, and *loser* slots, a similar pattern is seen, albeit with a smaller improvement among the *winner* and *nominee* slots. This difference in performance between the independent SVMs and Learning to Search methods can be explained by the global modeling performed by the Learning to Search algorithm. In the Learning to Search approach, predictions on the *party-winner* slot can consider how the *party* and *party-loser* slots have been filled thus far. Finding evidence for an entity belonging to the *party* slot can boost confidence in belonging to either of the child slots, while finding evidence for an entity belonging to either of the child slots can reduce support for membership in the sibling (e.g. an organization cannot be both a *party-winner* and *party-loser*).

To a lesser degree, we find that this pattern remains true when comparing the independent neural networks approach to Learning to Search. While independent neural networks show slightly stronger performance on the *winner* and *party-winner* slots, the Learning to Search approach continues to show massive improvement on the *loser* and *party-loser* slots compared to independent neural networks, as well as slight improvements on the *nominee* and *party* slots.

As a second example, let us now consider similar analysis on the *attacks* domain. The number of available training examples for each of the 6 slots for the *attack* macro-event type may be seen in Figure 7.3. As a whole, we find that this event type has a less varied distribution of examples over the different slots, with the exception of the *injured* slot, which has only a very small number of examples.

Detailed results from our top methods on the *attacks* domain may be seen in Figure 7.4. As was the case in *elections* with the *running-mate* categories, we find that none of these methods are able to identify arguments for slots with extremely low amounts of training data. The difficulty of the *injured* slot is further compounded by the semantic difficulty of the slot itself. In articles about attacks, typically the main focus is on the perpetrators or people who were killed. Injured survivors tend to receive less focus in texts than deceased victims, so it is much more difficult to accurately extract such arguments.

The strength of Learning to Search on arguments belonging to parent-child relationships is is seen once again in the *attacks* domain via the *dead* and *victim* slots. More generally, however, it is clear that Learning to Search provides a strong benefit across all of the person-type slots: *perpetrator*, *victim*, and *dead*. Intuitively, this makes sense because each of these slots have close semantic relations to each other. Perpetrators cannot be victims, and by extension cannot be dead victims. Any argument filling the *dead* slot must be included within the *victim* slot. By taking into account global information when filling each of these slots, we can ultimately produce a more accurate final prediction.

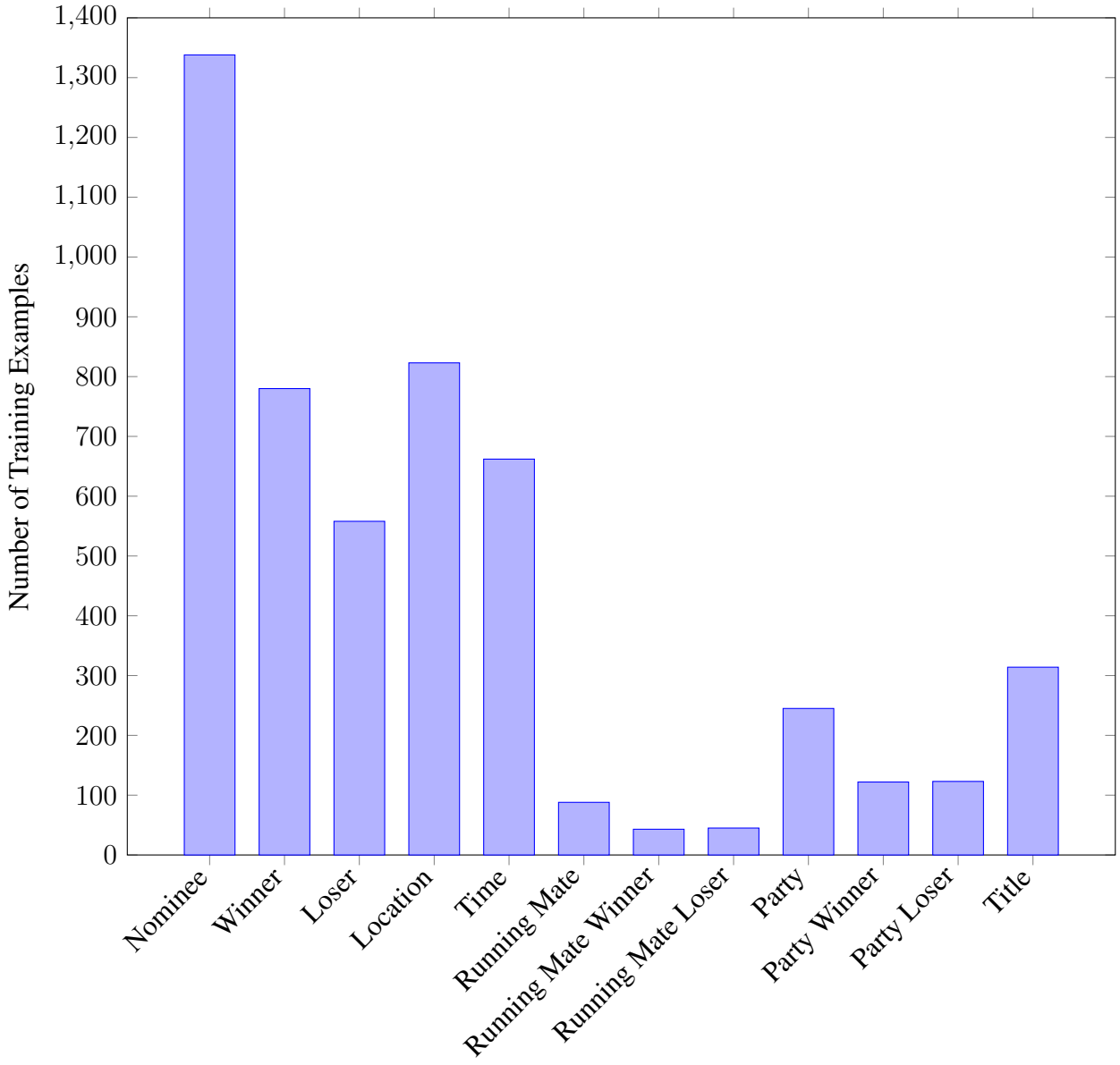


Figure 7.1: Number of training examples for each category in the *elections* domain.

7.6 Experiments with Limited Training Data

Our experiment results so far show relatively poor performance of the QA Neural Networks compared to even simple methods like independent SVMs. However, a notable strength of the algorithm is that it has no reliance whatsoever on having annotated training data for the target macro-event domains. This is particularly useful for scenarios where rapid deployment is desired for a new domain of interest, and little or no time at all is available to annotate documents for the target domain.

To illustrate this strength, we conduct experiments on the *attacks* training data where we

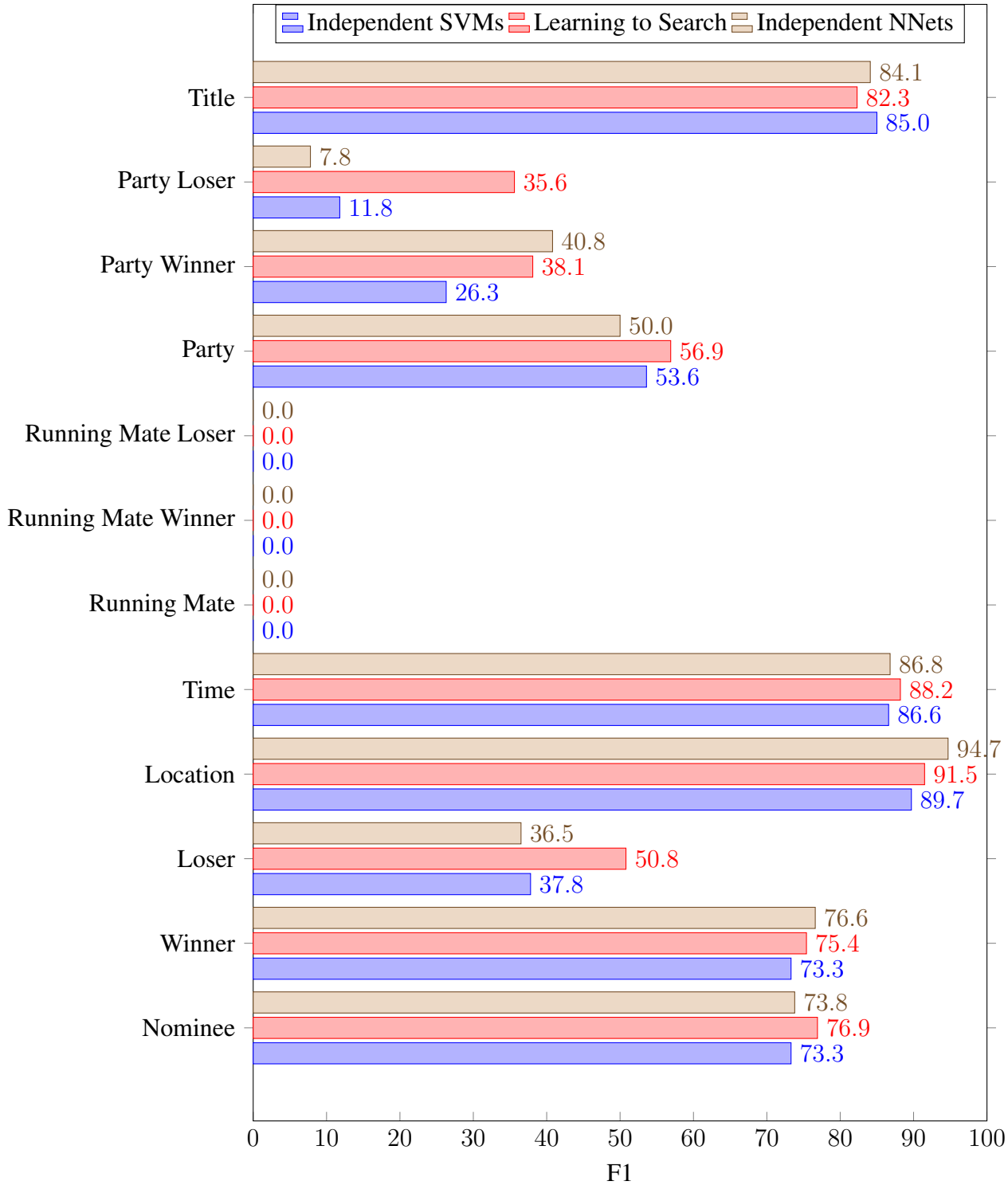


Figure 7.2: Performance by the top three methods on individual categories in the *elections* data.

vary the amount of training data made available to the machine learning algorithms, up to 100 documents. We compare performance from our four strongest methods: QA Neural Networks,

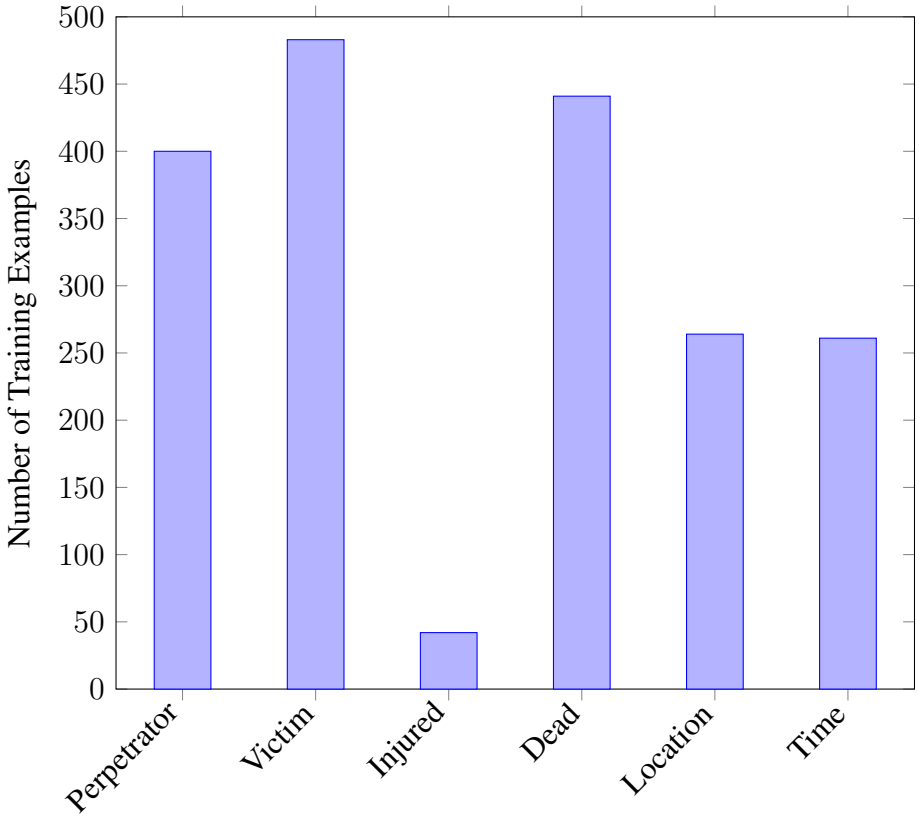


Figure 7.3: Number of training examples for each category in the *attacks* domain.

Independent SVMs, Learning to Search for Macro-Events, and Independent Neural Networks.

Results for micro-averaged F1 and macro-averaged F1 may be seen in Figures 7.5 and 7.6 respectively. On micro-averaged F1, we find that only the Learning to Search method can outperform the QA Neural Networks under this setting, and even then only when at least 90 documents of training data are made available to the system. The independent SVMs and independent neural networks are even weaker, and are completely unable to match the performance of the QA Neural Networks. When training data becomes extremely limited, the difference is made even more clear, with performance across all three of these methods dropping to abysmal levels of performance. In contrast the QA Neural Networks, having no reliance on macro-event training data, consistently achieve a micro-averaged F1 of 39.1.

On macro-averaged F1, we see a very similar situation, except this time none of the competing methods are able to exceed the QA Neural Networks. This once again illustrates the strength of the QA Neural Networks for domains with limited training data. The lower performance of the competing methods here also serve to emphasize the difficulty in training extractors for rare classes, as macro-averaged results give equal weight to both commonly seen argument roles (such as perpetrators) and rarely seen argument roles (such as injured victims).

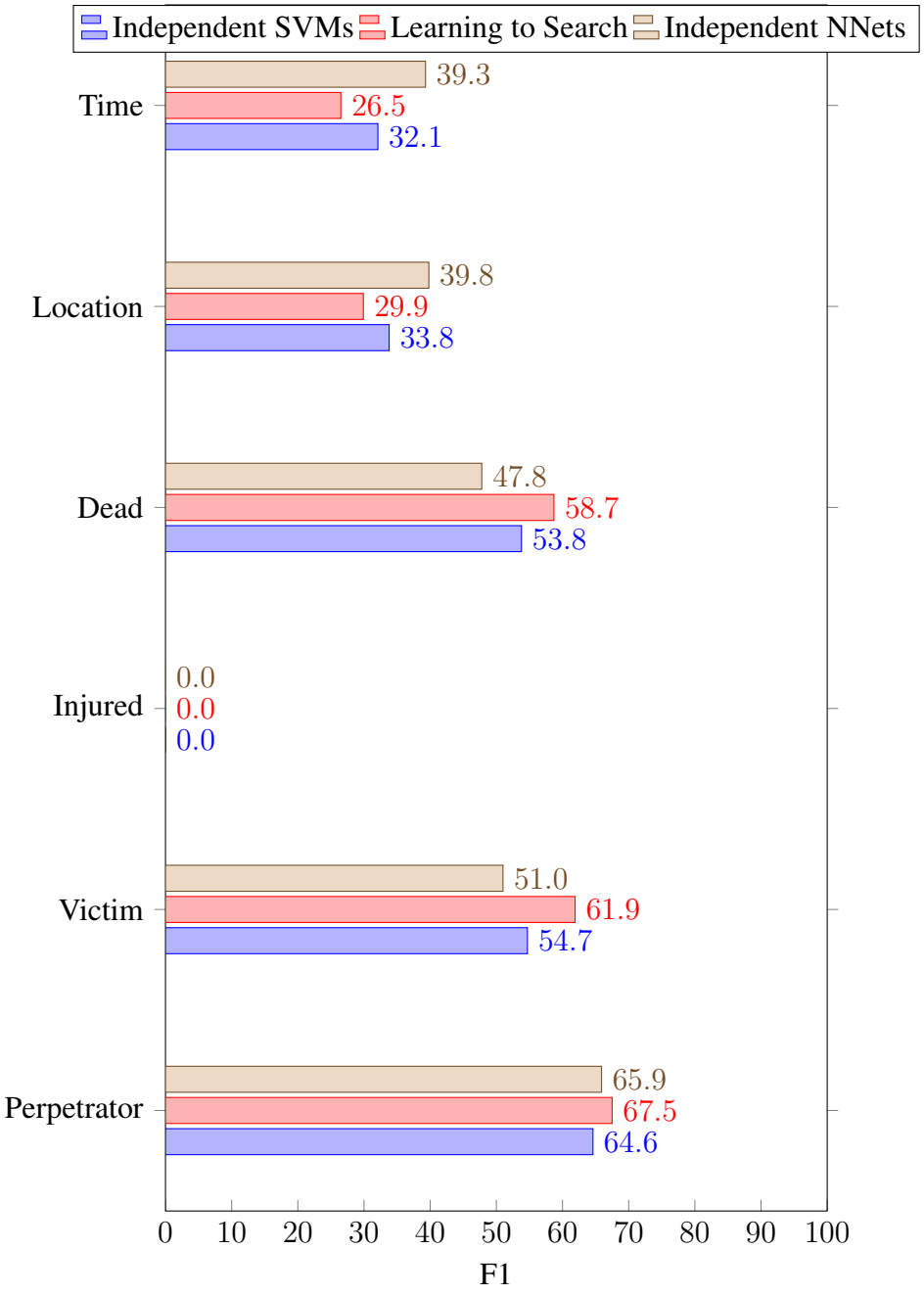


Figure 7.4: Performance by the top three methods on individual categories in the *attacks* data.

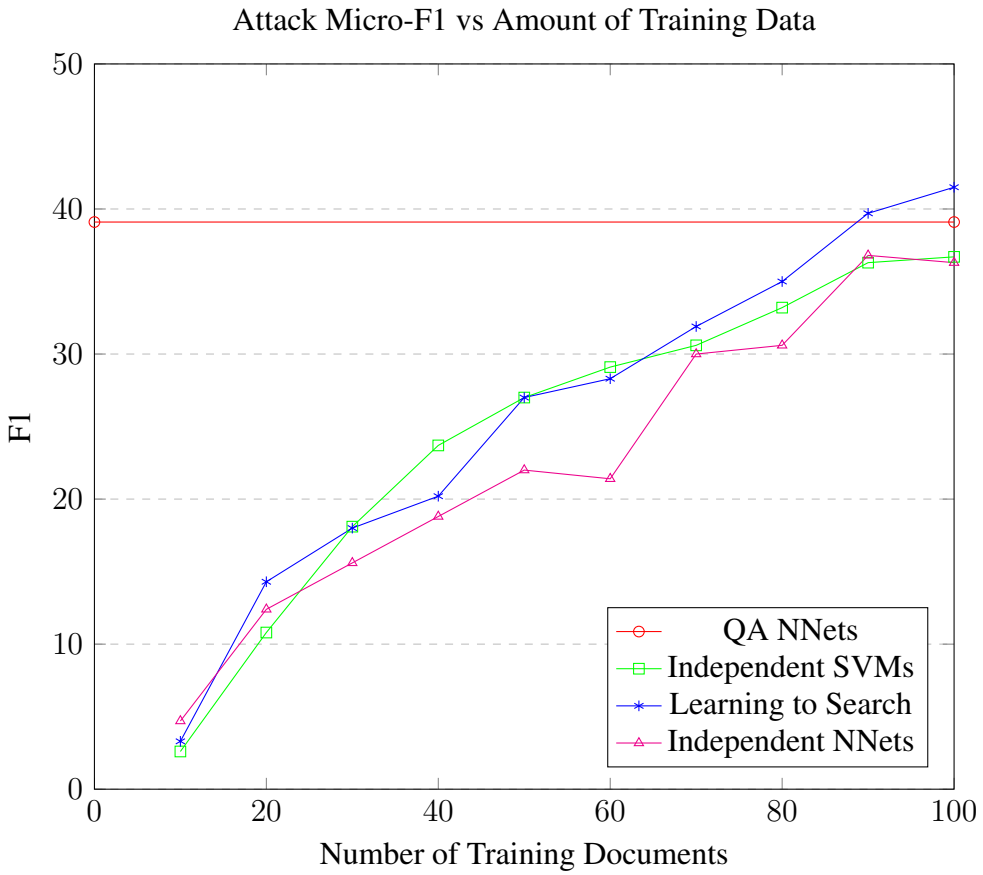


Figure 7.5: Micro-F1 results on the attack domain as the percentage of training data used varies.

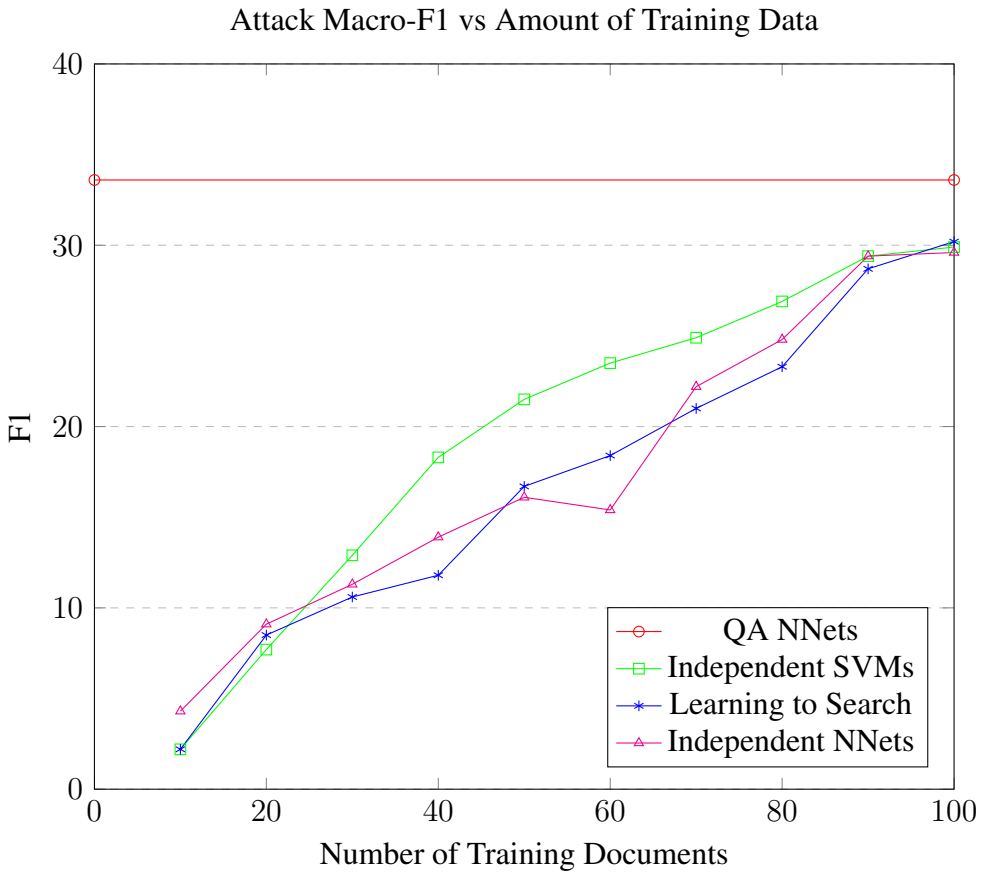


Figure 7.6: Macro-F1 results on the attack domain as the percentage of training data used varies.

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 three methods for addressing this problem: 1.) a structured model that jointly learns the argument slots of a macro-event, 2.) a feedforward neural network trained directly on annotated macro-event data, and 3.) a neural machine reading comprehension model that requires zero available training data in the target domain. Our experimental results show that structured prediction and feedforward neural models for this task can achieve significantly improved performance over independent SVMs and other baseline methods, while applying deep neural networks for machine reading comprehension can achieve high performance for domains with little available training data.

The overall contributions of this thesis advance the state-of-the-art for document-level event extraction in multiple ways. First, we introduce a number of algorithms for macro-event extraction which use much more sophisticated machine learning techniques that have been seen to date for document-level extraction. Second, we demonstrate the generalizability of these techniques to multiple general-purpose domains (as opposed to the extremely specific domains studied through MUC). Third, we demonstrate that these techniques are not limited to English language documents, as we have shown that such algorithms can be easily ported to different languages while retaining high quality results. Finally, we introduce the use of notion of applying deep machine reading comprehension models to the event extraction problem, which can allow for rapid deployment to new domains without any reliance whatsoever on target domain annotated training data.

We envision multiple promising directions in which future research can expand upon this work. One very natural extension would be to consider the problem of multi-document event extraction. The work contained in this thesis treats each document as a separate unique event. However, in the real world, events are often described in many different news articles, with each one potentially containing new information that was unseen in previous documents. This is particularly true for breaking news, where limited information may be available in the initial reporting document, with later documents containing more detailed, specific information. Future

work into macro-event extraction may wish to consider such cases by development of macro-event coreference across documents.

A further area for future development is in cross-lingual macro-event extraction. In this thesis, we have shown that our algorithms are easily capable of handling non-English texts, provided the availability of training data. However, we have not explored the idea of using training data from one language in order to boost performance in another language. Given the recent success of cross-lingual NLP for a variety of tasks, including part-of-speech tagging [24, 115], dependency parsing [1, 24, 87, 131], and named entity recognition [104], this could be another promising area of development for future research.

There are also promising research directions to explore outside of traditional news articles. One such direction could be to examine other genres of written text, including microblogs and other forms of social media. Even more interesting would be to explore related problems in audio and visual analysis. For example, consider the problem of applying macro-event extraction to broadcast news, rather than written news articles. Although the input format differs from what has been studied in this thesis, one could develop techniques for combining speech recognition techniques with macro-event extraction in order to create template-based summaries of broadcast news.

More ambitiously, one could consider a multimodal macro-event extraction system, which would build upon all of the aforementioned ideas. In the real-world, news is typically found via multiple sources – written news articles, social media posts, broadcast news audio, and live footage just to name a few. To truly understand the key details of a newsworthy event, one may need to analyze multiple sources of data, as any individual source may not contain all the necessary information (this is especially true in the case of breaking news, where additional information becomes available as the event is ongoing). A multimodal macro-event extraction system could be highly valuable to analysts seeking to understand complex, newsworthy events. To successfully do so however, would require advances in cross-document event extraction (to merge information across multiple documents), cross-lingual event extraction (documents may be written in different languages), audio event extraction (need to be able to understand spoken dialogue), and video event extraction (need to be able to understand footage of an event and identify key actors).

In addition to advances in multi-modal macro-event extraction, there is also room for future work on the machine learning techniques used for macro-event extraction. One such direction to explore is via the field of cross-domain transfer learning. As noted in this thesis, a major challenge for rapid deployment of macro-event extraction algorithms to new domains is the limited amount of available training data. We have shown that machine reading comprehension models can offer substantial benefits over directly supervised methods in such scenarios. However, the current approach for machine reading comprehension does not utilize any on-domain macro-event annotated data. In practice, even for cases of rapid deployment, there may still be some annotated data available, even if only on the order of dozens rather than hundreds or thousands of documents. Using this small amount of on-domain data to tune a general-purpose model toward the target domain may be able to result in additional boosts in performance.

Additional improvements may also be observed by exploring alternative classifiers for use in our Learning to Search algorithm. In our experiments, we utilized SVMs, however in general any classifier could be used as the base classifier of a Learning to Search algorithm. Using

other techniques for classification, such as directly supervised neural networks or even machine comprehension networks, could result in further improvements to macro-event extraction. In particular, an approach that could be useful is to use machine comprehension networks as the base classifier during the initial deployment to a new domain (when little data is likely to be available), and subsequently transition to more data-hungry models (such as Learning to Search) as new annotated data for the domain becomes available. The point of transition between models could be determined via approaches similar to DUAL (Dual Strategy Active Learning) [33], or one could use a weighted vote among various models, where the weights for the directly supervised methods start low and increase as annotated data becomes more readily available. Such approaches would allow extraction to be robust to both data-rich and data-poor scenarios.

Beyond these suggestions, ensemble techniques could prove useful in general for macro-event extraction. As seen in our experimental analysis, the best performing method for each individual argument slot often varies. The best overall performance is likely to be achieved by some combination of models (for example, applying Learning to Search for semantically-related slots, machine comprehension to slots with little or no training data, and independent neural networks for the remaining slots).

Bibliography

[1] Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. Many languages, one parser. In *TACL*, 2016. 8

[2] Douglas E Appelt, Jerry R Hobbs, John Bear, David Israel, and Mabry Tyson. Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*, 1993. 1.1, 2.1.1

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

[4] Stephanie E. August and Charles P. Dolan. Hughes research laboratories trainable text skimmer: Muc-4 test results and analysis. In *Proceedings of the 4th Conference on Message Understanding*, 1992. 1.1, 2.1.1

[5] Cosmin Adrian Bejan and Sanda Harabagiu. Unsupervised event coreference resolution. *Computational Linguistics*, 2014. 1.1

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

[7] Emanuela Boroş, Romaric Besançon, Olivier Ferret, and Brigitte Grau. Event role extraction using domain-relevant word representations. In *EMNLP*, 2014. 1.1, 2.1.1, 5.1

[8] 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, 2.3.1

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

[10] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011. 7.2

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

[12] 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

[13] Chen Chen and Vincent Ng. Joint modeling for chinese event extraction with rich linguistic features. In *COLING*, 2012. 2.1.3

[14] Chen Chen and Vincent Ng. Sinocoreferencer: An end-to-end chinese event coreference resolver. In *LREC*, 2014. 2.1.3

[15] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, 2014. 5.1

[16] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the cnn/daily mail reading comprehension task. In *ACL*, 2016. 2.2, 5.1

[17] 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. (document), 1.1, 2.1.2, 2.6

[18] Zheng Chen and Heng Ji. Language specific issue and feature exploration in chinese event extraction. In *HLT-NAACL*, 2009. 2.1.3

[19] 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. 2.1.3

[20] 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, 2.1.1

[21] 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

[22] Jason PC Chiu and Eric Nichols. Named entity recognition with bidirectional lstm-cnns. *arXiv preprint arXiv:1511.08308*, 2015. 5.1

[23] 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. 6.1

[24] Shay B. Cohen, Dipanjan Das, and Noah A. Smith. Unsupervised structure prediction with non-parallel multilingual guidance. In *EMNLP*, 2011. 8

[25] Ronan Collobert. Deep learning for efficient discriminative parsing. In *AISTATS*, 2011. 5.1

[26] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 2011. 5.1

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

[28] 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, 5.1

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

[30] Gerald DeJong. Prediction and substantiation: A new approach to natural language processing. *Cognitive Science*, 1979. 1.1, 2.1.1

[31] Bhuwan Dhingra, Hanxiao Liu, William W Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. In *arXiv preprint arXiv:1606.01549*, 2017. 2.2, 5.1, 6

[32] George R Doddington, Alexis Mitchell, Mark A Przybocki, Lance A Ramshaw, Stephanie Strassel, and Ralph M Weischedel. The automatic content extraction (ace) program-tasks, data, and evaluation. In *LREC*, 2004. 2.1.2

[33] Pinar Donmez, Jaime G Carbonell, and Paul N Bennett. Dual strategy active learning. In *ECML*, 2007. 8

[34] Michael Elhadad. *Using argumentation to control lexical choice: a functional unification implementation*. PhD thesis, Columbia University, 1993. 2.3.2

[35] 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, 2.3.1

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

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

[38] 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

[39] 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

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

[41] 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

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

[43] Ralph Grishman, David Westbrook, and Adam Meyers. Nyus english ace 2005 system description. In *Proc. ACE 2005 Evaluation Workshop*, 2005. 1.1, 2.1.2

[44] 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, 5.1, 6.2

[45] 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

[46] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading childrens books with explicit memory representations. In *ICLR*, 2016. 6.2

[47] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R.

Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012. 13

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

[49] Andrew Hsi, Jaime Carbonell, and Yiming Yang. Modeling event extraction via multilingual data sources. In *Text Analysis Conference (TAC 2015)*, 2015. 2.1.3

[50] Andrew Hsi, Jaime Carbonell, and Yiming Yang. Cmu_cs_event tac-kbp2016 event argument extraction system. In *Text Analysis Conference (TAC 2016)*, 2016. 2.1.3

[51] Andrew Hsi, Yiming Yang, Jaime Carbonell, and Ruochen Xu. Leveraging multilingual training for limited resource event extraction. In *COLING*, 2016. 2.1.3

[52] Ruihong Huang and Ellen Riloff. Peeling back the layers: detecting event role fillers in secondary contexts. In *ACL*, 2011. 1.1, 2.1.1

[53] Ruihong Huang and Ellen Riloff. Modeling textual cohesion for event extraction. In *AAAI*, 2012. 1.1, 2.1.1

[54] Scott B Huffman. Learning information extraction patterns from examples. In Stefan Wermter, Ellen Riloff, and Gabriele Scheler, editors, *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*. Springer-Verlag, Berlin, 1996. 1.1, 2.1.1

[55] 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, 2.1.3

[56] 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, 2.1.3

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

[58] 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

[59] Hongyan Jing and Kathleen R. McKeown. Cut and paste based text summarization. In *NAACL*, 2000. 1.1, 2.3, 2.3.2

[60] Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. Text understanding with the attention sum reader network. In *ACL*, 2016. 2.2, 5.1

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

[62] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *ACL*, 2014. 2.1.2

[63] Jun-Tae Kim and Dan I Moldovan. Acquisition of semantic patterns for information ex-

traction from corpora. *IEEE*, 1993. 1.1, 2.1.1

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 5.1

[65] Dustin Lange, Christoph Böhm, and Felix Naumann. *Extracting structured information from Wikipedia articles to populate infoboxes*. 2010. ISBN 978-3-86956-081-6. 3.3.1

[66] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998. 2.1.2

[67] 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

[68] 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

[69] Wendy Lehnert and Beth Sundheim. A performance evaluation of text-analysis technologies. *AI magazine*, 1991. 1.1, 2.1.1, 2.1.1

[70] Peifeng Li, Guodong Zhou, Qiaoming Zhu, and Libin Hou. Employing compositional semantics and discourse consistency in chinese event extraction. In *EMNLP*, 2012. 2.1.3

[71] Qi Li and Heng Ji. Incremental joint extraction of entity mentions and relations. In *ACL*, 2014. doi: 10.3115/v1/P14-1038. 7.1

[72] Qi Li, Heng Ji, and Liang Huang. Joint event extraction via structured prediction with global features. In *ACL*, 2013. 1.1, 2.1.2, 7.1

[73] 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, 2.1.2, 7.1

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

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

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

[77] 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

[78] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 1989. 2.1.2

[79] 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

[80] 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. 2.1.4

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

[82] 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. 2.1.4

[83] 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. 2.1.4

[84] 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. 7.2

[85] André FT Martins, Mario AT Figeuiredo, Pedro MQ Aguiar, Noah A Smith, and Eric P Xing. An augmented lagrangian approach to constrained map inference. In *ICML*, 2011. 2.1.2

[86] Ryan McDonald, Koby Crammer, and Fernando Pereira. Online large-margin training of dependency parsers. In *ACL*, 2005. 2.1.2

[87] Ryan McDonald, Slav Petrov, and Keith Hall. Multi-source transfer of delexicalized dependency parsers. In *EMNLP*, 2011. 8

[88] Kathleen R McKeown, Judith L Klavans, Vasileios Hatzivassiloglou, Regina Barzilay, and Eleazar Eskin. Towards multidocument summarization by reformulation: Progress and prospects. *AAAI*, 1999. 1.1, 2.3, 2.3.2

[89] Michele Merler, Bert Huang, Lexing Xie, Gang Hua, and Apostol Natsev. Semantic model vectors for complex video event recognition. *IEEE Transactions on Multimedia*, 2012. 2.1.4

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

[91] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *EMNLP*, 2004. 1.1, 2.3, 2.3.1

[92] Thien Huu Nguyen and Ralph Grishman. Event detection and domain adaptation with convolutional neural networks. In *ACL*, 2015. doi: 10.3115/v1/P15-2060. (document), 1.1, 2.1.2, 2.5, 2.1.2, 2.6

[93] 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, 6.2

[94] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999. 2.3.1

[95] Siddharth Patwardhan and Ellen Riloff. Learning domain-specific information extraction patterns from the web. In *Proceedings of the Workshop on Information Extraction beyond the Document*, 2006. 1.1, 2.1.1

[96] Siddharth Patwardhan and Ellen Riloff. Effective information extraction with semantic affinity patterns and relevant regions. In *EMNLP-CoNLL*, 2007. 1.1, 2.1.1

[97] 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, 2.1.1

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

[99] 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

[100] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*, 2016. 2.2, 6.2

[101] 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. 2.1.4

[102] Amjad Rehman and Tanzila Saba. Features extraction for soccer video semantic analysis: current achievements and remaining issues. *Artificial Intelligence Review*, 2014. 2.1.4

[103] 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

[104] Alexander E. Richman and Patrick Schone. Mining wiki resources for multilingual named entity recognition. In *ACL*, 2008. 8

[105] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, 1993. 1.1, 2.1.1

[106] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *AAAI*, 1996. 1.1, 2.1.1

[107] 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

[108] Jacques Robin. *Revision-based generation of Natural Language Summaries providing historical Background*. PhD thesis, Columbia University, 1994. 2.3.2

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

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

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

[112] Cicero D Santos and Bianca Zadrozny. Learning character-level representations for part-of-speech tagging. In *ICML*, 2014. 5.1

[113] Cicero Nogueira dos Santos and Victor Guimaraes. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008*, 2015. 5.1

[114] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-

scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 5.1

[115] Benjamin Snyder, Tahira Naseem, Jacob Eisenstein, and Regina Barzilay. Adding more languages improves unsupervised multilingual part-of-speech tagging: A bayesian non-parametric approach. In *NAACL*, 2009. 8

[116] Stephen Soderland, David Fisher, Jonathan Aseltine, and Wendy Lehnert. Crystal: Inducing a conceptual dictionary. In *IJCAI*, 1995. 1.1, 2.1.1

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

[118] Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. Automatic pattern acquisition for japanese information extraction. In *HLT*, 2001. 1.1, 2.1.1

[119] Kiyoshi Sudo, Satoshi Sekine, and Ralph Grishman. An improved extraction pattern representation model for automatic ie pattern acquisition. In *ACL*, 2003. 1.1, 2.1.1

[120] 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. 2.1.4

[121] Beth Sundheim, editor. *Fourth Message Understanding Conference (MUC-4)*, 1992. Morgan Kaufmann Publishers. 1.1, 2.1.1

[122] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998. 4.1

[123] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015. 5.1

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

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

[126] 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 Recognition (CVPR)*, 2015. 2.1.4

[127] 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. 2.1.4

[128] Bishan Yang and Tom Mitchell. Joint extraction of events and entities within a document context. In *NAACL*, 2016. doi: 10.18653/v1/N16-1033. (document), 1.1, 2.1.2, 2.4

[129] Roman Yangarber. Counter-training in discovery of semantic patterns. In *ACL*, 2003. 1.1, 2.1.1

[130] 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, 2.1.1

[131] Daniel Zeman and Philip Resnik. Cross-language parser adaptation between related languages. In *IJCNLP*, 2008. 8

[132] Xiaoqing Zheng, Hanyang Chen, and Tianyu Xu. Deep learning for chinese word segmentation and pos tagging. In *EMNLP*, pages 647–657, 2013. 5.1